Haaga-Helia
ammattikorkeakoulu Oy

Päiväkirjaopinnäytetyö front-end -kehittäjän arjesta

Aleksanteri Rytkönen

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2018

Haaga-Helia ammattikorkeakoulu Oy

Tiivistelmä

22.5.2018

Tekijä(t)

Aleksanteri Rytkönen

Koulutusohjelma

Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko

Päiväkirjaopinnäytetyö front-end -kehittäjän arjesta

Sivu- ja liite-sivumäärä

45

Opinnäytetyön otsikko englanniksi

Thesis report journal of front-end developer’s daily life

Tämä opinnäytetyö on päiväkirjaopinnäytetyö, missä seurataan front-end kehittäjän päivittäisiä työtehtäviä. Seurantajakso on 10 viikkoa, minkä ajalta kuvaillaan päivittäisiä työtehtäviä. Työviikkojen päätteeksi kirjoitetaan viikkoanalyysi, missä käydään läpi kohdattuja ongelmia ja ratkaisuja, sekä peilataan niitä ammattikirjallisuuteen.

Ennen viikkoseurannan aloittamista käydään läpi kehittäjän nykyinen toimenkuva, vaadittavat taidot ja sidosryhmät. Viikkoseurannan jälkeisellä viikolla käydään läpi viikkoseurannan aikana tehtyjä havaintoja ja osaamisen kehitystä ja peilataan sitä aloitusviikon tilanteeseen.

Kirjoittaja toimii front-end kehittäjänä pienyrityksessä. Pääasiallisena toimenkuvana on kehitystyö kiinteistön kylmäseurantaan keskittyvän ohjelmiston parissa. Pienemmässä roolissa on myös vähän vanhempien järjestelmien modernisointi vastaamaan paremmin nykyaikaa.

Seurantajakson lopussa analysoidaan kirjoittajan kehitystä näiden viikkojen aikana ja käydään läpi uusia menetelmiä ja huomioita työhön liittyen. Seurantaviikkojen aikana kirjoittajan toimenkuva ja osaaminen laajeni monella eri osa-alueella.

Asiasanat

Ohjelmistokehitys, Angular, JavaScript, jQuery

Sisällys

1 Johdanto

Aloitin maaliskuussa 2017 työskentelyn Core Factory Oy:ssä front-end -kehittäjänä. Työllistyin heti täysipäiväiseksi kehittäjäksi, aloittaen aluksi pienemmistä ja yksinkertaisemmista muutoksista. Päästyäni paremmin sisälle työtapoihin, sekä tekniikoihin olen saanut haasteekseni yhä monimutkaisempia ja haastavampia toteutuksia.

Tämän opinnäytetyön tarkoitus on tehdä päiväkirjaa 10 viikon ajalta liittyen työtehtäviini. Jokaisen viikon päätteeksi on tarkoitus muodostaa kuva viikon tehtävistä, ratkaisuista ja mahdollisista ongelmista ja peilata näitä aiheisiin liittyviä kirjallisuuslähteitä vasten. Aikaväli opinnäytetyölle on 05.02.2018 – 05.02.2018.

Core Factory on kiinteistöjen älyteknologiaan erikoistunut suomalainen laitevalmistaja. Suunnittelemme ja valmistamme tuotteita, jotka yhdistävät kiinteistöjen laitteet käyttäjäystävälliseksi kokonaisuudeksi. Tavoitteemme on luoda työtä, sähköä ja rahaa säästäviä älykkäitä ratkaisuja mm. yrityksiin, oppilaitoksiin ja virastoihin.

Työtehtäviini kuuluu pääasiassa front-end -kehitys, sekä erilaiset pienemmät tehtävät tarpeen niin vaatiessa. Suurimman osan ajasta teen uusia komponentteja front-end ohjelmaamme, sekä integroin ne yhteensopiviksi taustajärjestelmiemme kanssa. Yleensä asiakkaalla on tarve johonkin tiettyyn ominaisuuteen, joka välitetään esimiehiltä graafikolle. Graafikon hyväksytettyä oma näkemyksensä siirtyy kyseisen toiminnallisuuden tekeminen minulle. Tehtäviini kuuluu myöskin ohjelmiston testaaminen ja siinä mahdollisesti ilmenevien virheiden korjaaminen.

2 Lähtötilanteen kuvaus

Olen työskennellyt Core Factoryllä nyt 11 kuukautta ja työtehtäviini on pääasiallisesti kuulunut Optiwise kylmävalvontasovelluksen front-endin kehitys. Sovelluksen taustalla toimii ZeeCore laite, joka on liitettynä esimerkiksi kylmäjärjestelmiin ja kerää niistä dataa ZeeCloud pilvipalveluun. Optiwise sovelluksen on tarkoitus näyttää järjestelmistä kerättyä dataa ja tehdä siitä koosteita ja visualisointeja, jotta käyttäjien olisi helppo huomata mahdolliset vikatilanteet ja puuttua niihin nopeasti.

2.1 Oman nykyisen työn analyysi

Työskentelen n. 5 hengen tiimissä, missä pääasiallinen front-end -kehitys on minun vastuullani. Työtehtäväni liittyvät suurimmaksi osaksi Optiwise sovelluksen front-endin -kehittämiseen. Optiwise sovellus on front-endin osalta rakennettu Angular frameworkillä.

Työtehtäviini kuuluvat seuraavat osa-alueet:

1. Uusien komponenttien toteuttaminen ja suunnittelu annettujen määrittelyjen mukaan.
    - Komponenttien toteuttamisen yleiset työvaiheet:
      Asiakas tilaa uuden ominaisuuden, mistä graafikkomme tekee layoutin, minkä pohjalta alan suunnittelemaan komponenttia. Tilattu komponentti voi olla esimerkiksi datan visualisointi työkalu energiakulutuksen seurantaan.
2. Aktiivinen tiedon keruu.
3. Suunnittelupalavereihin osallistuminen.
4. Vanhojen komponenttien päivitys / korjaaminen.
5. Sovelluksessa olevien kirjastojen päivitys

Työtehtävissäni vaadittava osaaminen:

1. Angular
2. Javascript
3. Typescript
4. HTML5
5. CSS3
6. Git
7. REST-rajapinnat

Olen työtehtävissäni kehittynyt paljon, varsinkin Angular sovelluskehyksen ja Javascript (ECMAScript 6) käytössä. Kehitykseni on tapahtunut paljolti hakemalla internetistä tietoa ja lukemalla esimerkiksi Angularin dokumentaatiota. Kollegoiden apu on myöskin ollut

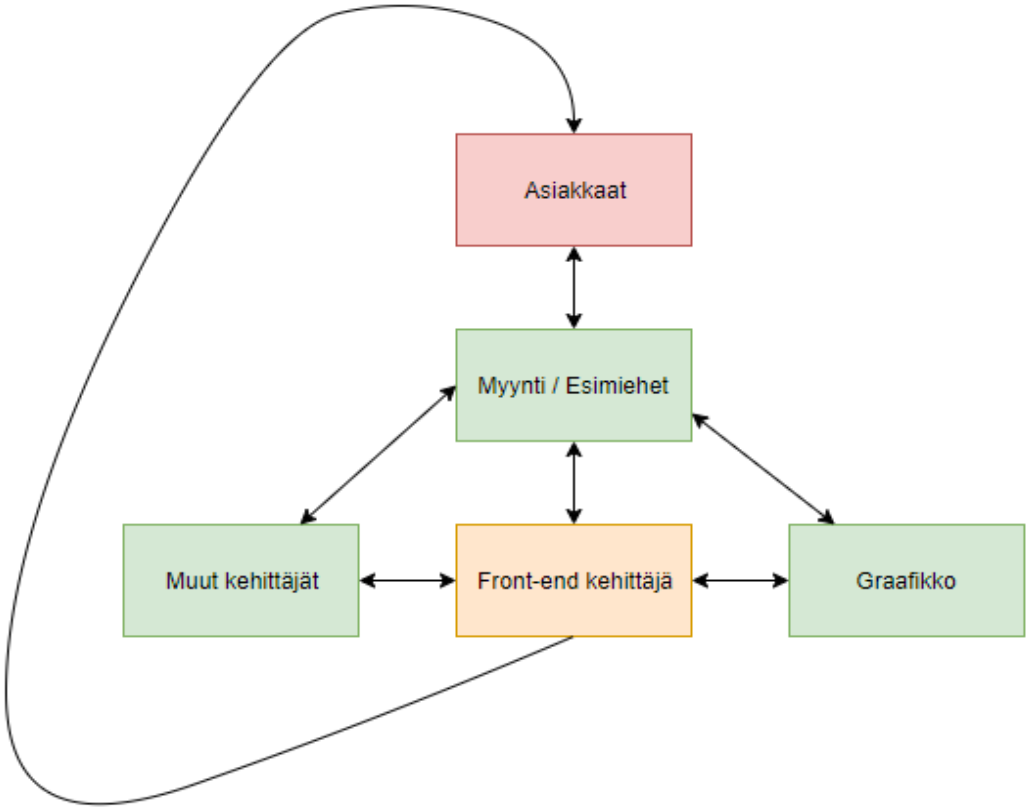
korvaamatonta, kun vastaan on tullut ongelmia mihin en ole itse löytänyt ratkaisua. Tärkein taito työtehtävissäni selviytymiseen on varmastikin tiedon hankinta. Kun pitää luoda uusia komponentteja ja käyttää kirjastoja, mitä ei ole ennen tullut vastaan. Tällaisissa tilanteissa on hyvin tärkeää osata etsiä tietoa oikeista lähteistä ja soveltaa löytynyttä tietoa sopimaan omiin käyttötarkoituksiin.

Osaamisen taso suhteessa työtehtäviin:

1. Angular viitekehys
    - koen olevani taitava suoriutuja, ymmärrän Angular sovelluksen rakenteen ja pystyn toteuttamaan itsenäisesti uusia vaadittavia komponentteja, sekä myöskin neuvomaan muita tarvittaessa.
2. Javascript
    - Javascriptin saralla koen myöskin olevani taitava suoriutuja. Osaan toteuttaa itsenäisesti tarvittavia toiminnallisuuksia sen avulla. Parannettavaa olisi vielä Javascriptin uudempien versioiden (ECMAScript 6) erilaisten toiminnallisuuksien käytössä.
3. Typescript
    - Typescript ja Javascript kulkevat pitkälti käsi kädessä. Typescript on Javascriptin supersetti ja lisää Javascriptiin muista ohjelmointikielistä tuttuja ominaisuuksia, kuten luokat ja staattisen tyypityksen.
4. Git
    - Gitin peruskäytön osaan jo hyvin ja selviydyn itsenäisesti sen käytöstä. Uusien haarojen luominen ja yhdistäminen olemassa oleviin haaroihin on minulle arkipäivää. Selviydyn myöskin mahdollisista merge conflict ongelmista itsenäisesti.
5. HTML5 & CSS3
    - Molemmat näistä ovat olleet minulle tuttuja jo pitkään ja koen olevani kokenut osaaja näiden saralla. Olen käyttänyt molempia jo useita vuosia ja pystyn neuvomaan muita myös uudempien ominaisuuksien kuten CSS Grid Layoutin käytössä.
6. REST-rajapinnat
    - Osaan hakea ja lähettää tietoa REST-rajapinnoista sujuvasti, mutta kehittämistä vaatisi vielä backend osaaminen ja itse rajapintojen luominen.

2.2 Sidosryhmät

Alla olevassa kaaviossa (kuvio 1) on kuvattu omaan työhöni eniten vaikuttavat sidosryhmät.



Kuvio 1. Sidosryhmäkaavio

Yrityksen sidosryhmät ovat omalta kannaltani melko selkeät. Olen ainoa pääasiallinen front-end kehittäjä ja teen eniten yhteistyötä graafikon ja muiden kehittäjiemme kanssa. Esimiehet kommunikoivat asiakasrajapinnassa ja tuovat asiakkaiden pyyntöjä eteenpäin. Yleensä esimiehiltä tulee graafikolle pyyntö luoda ulkoasuja sovelluksille. Kun ulkoasu on hyväksytty asiakkaan puolelta, siirtyy työ itse kehittäjille. Minun tehtäväni on pyrkiä kommunikoimaan graafikon kanssa tavoitellusta ulkoasusta ja sen toiminnallisuudesta, sekä keskustella muiden kehittäjien kanssa eri rajapinnoista ja toiminnallisuudesta mitä sovellukset tarvitsevat. Minun työni taas on se, mikä näkyy asiakkaille. Pystyn siis hyvin keskittymään omaan työhöni pelkässä kehittämisessä, koska muut ihmiset ovat vastuussa itse asiakasrajapinnassa toimimisessa.

3 Päiväkirjaraportointi

3.1 Seurantaviikko 1

Maanantai 05.02.2018

Maanantain tavoitteena oli jatkaa optiwise sovellukseen tehtävän dashboard moduuliin ensimmäistä osiota, joka sisälsi kulutusvertailun.

Olin edellisellä viikolla luonut pohjan valmiiksi itse moduulille ja vähän tehnyt kulutusvertailu komponenttia, mutta toteutus menikin suunnittelupalaverin jälkeen vähän uusiksi, joten lähdin aika puhtaalta pohjalta tekemään komponenttia. Sain graafikolta uuden layoutin, siitä miltä komponentin pitäisi näyttää. Tarkoituksena oli siis tehdä komponentti, missä on kymmenen parasta/huonointa kohdetta, joita voi vertailla itse valitsemallaan aikavälillä ja valita itse mistä ryhmästä kohteita valitaan. Kohteita tuli pystyä vertaamaan myöskin kolmen eri kulutuksen ja muutaman muun parametrin perusteella.

Tein ensimmäiseksi itse layoutin oikean näköiseksi, ilman mitään oikeaa toiminnallisuutta. Layoutissa on paljon värillisiä palkkeja, joiden toteuttamiseksi oli onneksi jo aikaisemmin toteutettu komponentti. Importoimalla kyseisen komponentin sai palkit toteutettua suhteellisen ongelmattomasti. Pieniä CSS muutoksia jouduin toki tekemään, liittyen palkkien väreihin ja kokoon, koska ne olivat erilaiset verrattuna toiseen komponenttiin missä vastaavan tyylisiä palkkeja käytettiin.

Päivän tavoitteena oli jatkaa kulutusvertailu komponentin tekemistä ja pääsin siinä ihan hyvälle mallille. Sain ulkoasun näyttämään suurin piirtein samalta, kuin graafikon antamassa layoutissa. Oikeaa dataa siinä ei toki vielä ollut, mutta siitä on hyvä jatkaa huomenna.

Tiistai 06.02.2018

Päivän tavoitteena oli jatkaa / yrittää saada valmiiksi kulutusvertailu komponentti.

Eilen sain komponentin ulkoasun valmiiksi, tänään oli siis aika saada siihen oikeaa dataa näkyviin. Backend puolella onneksi kaikki oli jo valmiina, koska meillä on toinen vastaavanlainen komponentti jo tehtynä, joten kaikki tarvittavat rajapinnat olivat jo olemassa. Lähdin siis toteuttamaan funktioita jotka käyttävät kyseisiä rajapintoja ja kyselevät dataa käyttäjän antamilla parametreillä. Datan haku onnistui, mutta pian huomasin ongelman.

Tarkoituksena oli hakea 10 parasta / huonointa kohdetta ja verrata niiden arvoja samaan aikaväliin vuosi aiemmin. Tämä tarkoitti sitä, että jouduin tekemään kaksi eri kyselyä, yksi nykyiselle datalle ja yksi vuodentakaiselle. Ongelmaksi tässä muodostui se, että vuosi takaperin 10 parasta kohdetta saattoi olla aivan eri kohteet, kuin ne ovat nyt. Ratkaisu tähän oli tietysti ensin hakea nykyiset kymmenen kohdetta, ottaa talteen niitten id:t ja hakea niillä vuoden takaiset vertailudatat.

Siirsin tekemäni muutokset staging ympäristöön ja testasin miltä ne näyttävät oikealla datalla. Tässä ilmeni sellainen ongelma, että dashboardin kulutusvertailu komponentin data ei vastannut sitä dataa, mikä näkyi täysimittaisessa kulutusvertailukomponentissamme. Syyn selvittäminen tähän ongelmaan jää kuitenkin huomiselle.

En saanut komponenttia aivan valmiiksi, mutta edistyin sen tekemisessä kuitenkin hyvin, joten päivästä jäi hyvät fiilikset.

Keskiviikko 07.02.2018

Päivän tavoitteena oli saada kulutusvertailu komponentti valmiiksi ja alkaa selvittämään miten saisin seuraavan komponentin toteutettua. Seuraavaksi komponentiksi oli tilattu kartta, missä tulisi näkyä kaikkien liikkeiden sijainnit Suomen kartalla.

Jatkoin siis siitä mihin eilen jäin, eli ongelman selvittämisestä miksi data kahden eri kulutusvertailu komponentin välillä ei vastannut toisiaan. Syy johtui siitä, että backendistä saattoi tulla enemmän, kuin kymmenen kohdetta, mutta koska vain kymmenen parasta/huonointa visualisoitiin ei data välttämättä näkynyt oikein. Ratkaisuksi tähän tein funktion mikä järjesteli arvot suurimmasta pienimpään tai pienimmästä suurimpaan, riippuen haettiinko kymmenen parasta vai huonointa kohdetta. Järjestelyn jälkeen käytin vielä Javascriptin slice funktiota ja leikkasin ryhmän sisältämään vain 10 ylintä arvoa. Tämän muutoksen jälkeen kulutusvertailu komponentti toimi niin kuin pitikin.

Loppupäivän käytinkin siihen, että selvitin, mitä olemassa olevia karttakirjastoja angulariin löytyisi. Päädyin nopealla selvityksellä kahteen potentiaaliseen kirjastoon, ngx-openlayers ja ngx-leaflet kirjastoihin. Torstaiksi jäisi molempien testaaminen ja päättäminen kumpi otetaan käyttöön.

Sain kulutusvertailu komponentin valmiiksi ja pääsin aloittamaan selvityksen siitä, mikä karttakirjasto sopisi parhaiten käyttöömme, joten päivä sujui hyvin.

Torstai 08.02.2018

Torstain tavoitteena oli testata, kumman kartta kirjaston ottaisin käyttöön ja aloittaa kartta komponentin toteuttaminen.

Aloitin testaamisen tekemällä kaksi uutta angular projektia, toiseen asensin ngx-openlayers kirjaston ja toiseen ngx-leaflet kirjaston. Molemmat kirjastoista oli helppo ottaa käyttöön, mutta en pitänyt ngx-openlayersin srid formaatista koordinaateissa. ngx-openlayersin dokumentaatio githubissa oli myöskin hyvin pieni verrattuna ngx-leaflet kirjaston dokumentaatioon. Valinnaksi tuli siis ngx-leaflet, minkä oma dokumentaatio oli melko hyvä ja itse leaflet kirjasto, mistä se on portattu sisältää erittäin hyvän dokumentaation.

Asensin ngx-leafletin optiwise projektiin ja tein dashboard moduulin uuden komponentin mihin aloin karttaa rakentamaan. Ensimmäinen ongelma oli, kun kartta tuntui hajoavan ja leviävän yli sille asetetuista rajoista gridissä. Kyseiseen ongelmaan löytyi kyllä ratkaisu, kun jaksoi vain selata ngx-leafletin dokumentaatiota vähän syvemmälle. Olin unohtanut asettaa polun leaflet.css tiedostoon angular-cli.json tiedoston sisälle. Tällä muutoksella kartta alkoikin näyttää siltä, kuin pitikin. Seuraavaksi piti tehdä rajapinta kutsu kartalle tulevien kohteiden sijainnista ja sijoittaa kohteet kartalle niille kuuluviin koordinaatteihin. Kohteet sai piirrettyä ongelmitta kartalle komponentin ensilatauksella, mutta kohteitten filtteröinti tuotti ongelmia. En saanut uusia näytettäviä kohteita päivittymään kartalle. Ei auttanut, kuin lähteä taas tutkimaan ngx-leafletin dokumentaatiota. Dokumentaatiosta selvisikin, että tulisi käyttää leafletMapReady output bindingia saadakseni referenssin itse kartta instanssiin. Tätä hyväksikäyttämällä filttereiden vaihtaminen ja kartan päivittäminen onnistuivatkin hyvin.

Lähdin päivään tavoitteena selvittää, mitä kartta kirjastoa käyttäisin ja aloittaa kartta komponentin teko. Onnistuin molemmissa tavoitteissa omasta mielestäni hyvin.

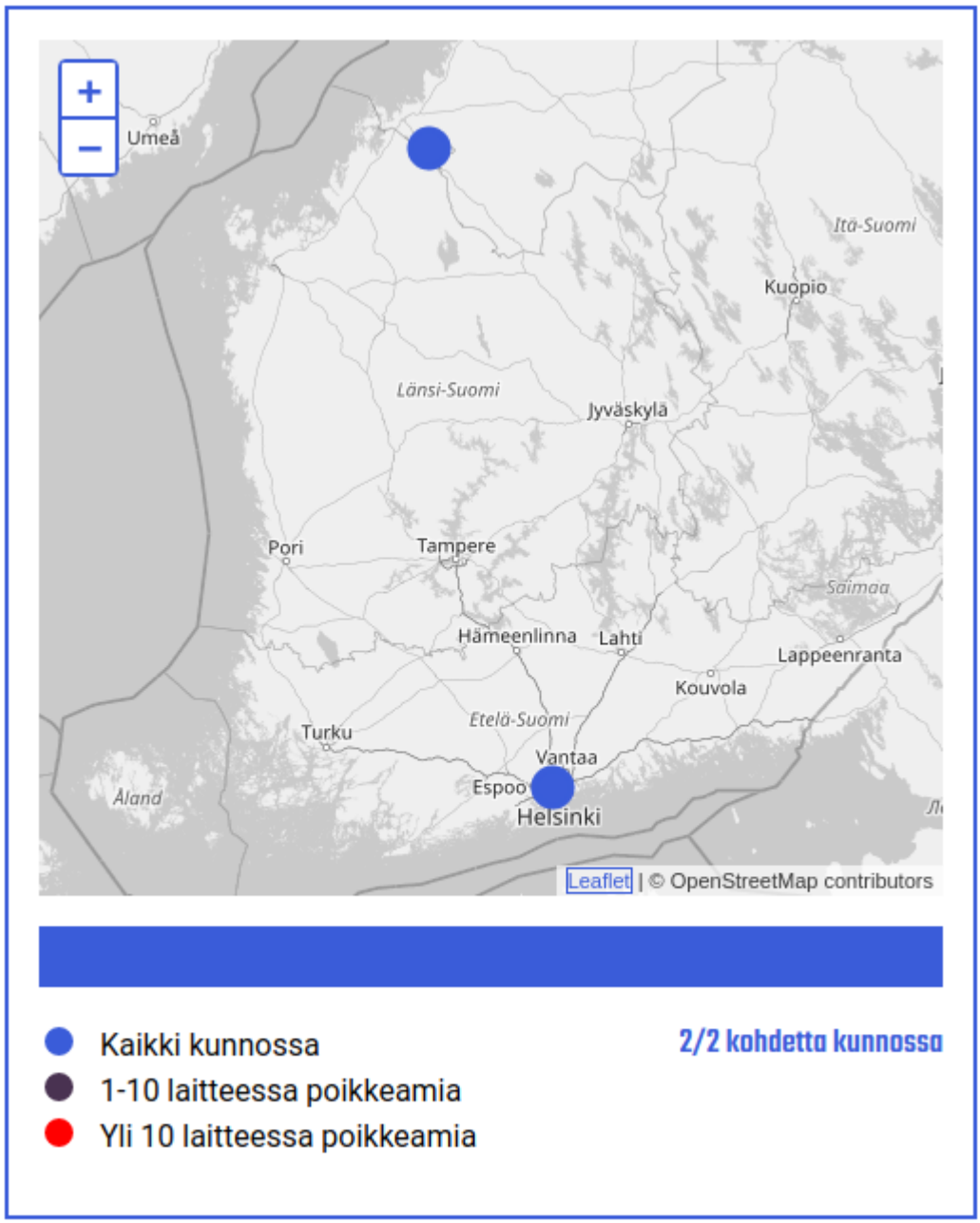
Perjantai 09.02.2018

Päivän tavoitteena saada kartta komponentti suurimmaksi osaksi valmiiksi.

Kartta komponentista uupui vielä kohteiden väritys, mikä indikoisi mikä on kunkin kohteen tila sillä hetkellä. Eli käytännössä tarkoituksena oli saada kartalla olevat kohteet näkymään kolmella eri värillä, riippuen siitä onko kaikki kunnossa, vähän vikoja vai paljon vikoja. Tämän toteuttaminen ei tuottanut erityisempiä ongelmia. Kartan alle piti myös toteuttaa palkki missä on kolmea eri väriä ja sen avulla pystyisi yhdellä silmäyksellä näkemään

kuinka suuressa osassa kohteista on jotain vikaa. Tämän ajattelin toteuttaa käytössämme olevan ng-bootstrap kirjaston progressbar komponentin avulla. Ikävikseni jouduin toteamaan, että ng-bootstrap kirjastoon ei ollut portattu bootstrapista löytyvää multiple bars (useampi palkki yhdessä palkissa) komponenttia. Joten jouduin toteuttamaan kyseisen toiminnallisuuden ihan normaalilla bootstrapilla. Onneksi kyseessä oli suhteellisen pieni toiminnallisuus, niin se ei vaatinut liikaa koodia.

Tämän jälkeen lisäsin vielä karttaan, joka näkyy kuviossa 2 automaattisen keskityksen näytettävissä olevien kohteiden perusteella. Tätä varten jouduin käyttämään leafletista löytyvää fitBounds funktiota, minkä käyttö oli kuitenkin helppoa.



Kuvio 2. Dashboard karttakomponentti

Päivän tavoitteena oli saada kartta ainakin lähes valmiiksi ja siinä onnistuin. Karttaan mahdollisesti tulee muutoksia tai lisää toiminnallisuuksia, kunhan ensimmäinen versio käy asiakkaalla arvioitavana.

Viikkoanalyysi

Viikko oli työntäyteinen ja pääsin toteuttamaan paljon uusia asioita. Kaikkein mielenkiintoisimpana oli karttakomponentin toteutus. En ollut ennen tehnyt mitään vastaavaa ja vaikka sen toteuttamiseen olikin annettu suhteellisen pitkä aikataulu, en alkuun ollut varma pystyisinkö sitä toteuttamaan. Onnekseni valmiita kartta kirjastoja löytyi useampia ja ainakin ngx-leaflet kirjasto mihin päädyin oli suhteellisen helppokäyttöinen.

Viikon suurin ongelma oli varmastikin saada kymmeneen parhaimpaan / huonoimpaan kohteeseen liittyvä ongelma kuntoon. Koska listat näyttivät erilaiselta, kuin niiden pitäisi näyttää jouduin käyttämään ongelman selvittämiseen kohtalaisen paljon aikaa. Joskus virheiden metsästäminen koodista on hyvinkin työlästä, onneksi nykyajan selaimissa on käytössä hyvät työkalut sitä varten. Resigin, Bibeaultin & Marasin (2016, 691) mukaan sovelluskehityksessä suuri osa ajasta kuluu virheiden korjaamisessa. Vaikka tämä voi joskus olla mielenkiintoista. Vähän kuin ratkaisisit suurempaakin mysteeriä, niin yleensä haluamme koodimme toimivan oikein ja ilman virheitä niin nopeasti, kuin mahdollista.

Käytän erittäin paljon Javascriptin loggaus toiminnallisuutta, kun yritän selvittää, miksi koodi käyttäytyy oudosti. Reisigin, ym (2016, 691) mukaan logitusta käytetään viestien tulostamiseen ohjelman suorituksen aikana, vaikuttamatta ohjelman toimintaan. Käytän logitusta myös paljon uusien komponenttien alkuvaiheessa, saadakseni selville toimiiko tietyt funktiot, niin kuin haluan tai ylipäätään tarkistaakseni tuleeko virheitä.

Oli myös erittäin hienoa saada pitkästä aikaa tehtyä täysin uutta toiminnallisuutta, ilman legacy koodin tuottamaa painolastia.

3.2 Seurantaviikko 2

Maanantai 12.02.2018

Heti maanantaipäivän alkaessa, minulla ei vielä ollut tiedossa mitä päivä tuo tullessaan, mutta töihin päästyäni sain heti alkuun tehtäväkseni päivittää ZeeCloud pilviportaalimme ulkoasua. Kyseessä on jo vanhempi sovellus (Vuodelta 2012) ja tämä onkin ensimmäinen

kerta, kun olen tekemisissä kyseisen sovelluksen kanssa. Sovellus on tehty Djangolla ja sovelluksen front-end käyttäen Django templateja. Ensimmäiseksi jouduin kuitenkin pystyttämään kehitysympäristön sovellusta varten. Kehitysympäristön pystyttäminen oli kuitenkin nopea operaatio, koska se ei vaatinut muuta kuin docker kontin ylösajon (docker-compose up). Jonka jälkeen piti vielä ajaa kontin sisällä Djangoon liittyvä komento sovelluksen ylläpito käyttäjän luomiseksi (python manage.py createsuperuser). Ainoa ongelma mikä tuli vastaan kehityksen aloituksen jälkeen, kun tekemäni muutokset CSS tiedostoon eivät ilmestyneet ollenkaan näkyviin. Kysyin kollegalta apua ongelman ratkaisemiseen ja ilmeni että docker kontin asetus tiedostoihin oli jäänyt rivi mikä käyttikin tuotanto, eikä kehitys ympäristön asetuksia.

En koskaan aiemmin ollut tehnyt mitään Djangolla, mutta sen käyttämä template moottori on hyvin yksinkertainen ja pelkän HTML:n ymmärryksellä pääsee jo pitkälle. Loppu päivä menikin graafikon antaman layoutin tutkimisessa ja ylipäätään pilviportaalin sivusto rakenteeseen tutustuessa.

Tiistai 13.02.2018

Tiistain tavoitteena olikin saada kunnolla tehtyä muutoksia pilviportaalimme ulkoasuun graafikon antaman layoutin pohjalta. Tarkoituksena oli uudistaa sivuston CSS tyylien rakenne ja päivittää vanhahtavilla tavoilla tehtyä CSS:ää modernimpaan suuntaan ja samalla uudistaa tietysti sivuston ulkoasukin modernimmaksi. Poistin paljon inline CSS muotoiluja, sekä muutenkin paljon id pohjaisia yksittäisiä muotoiluja. Nämä poistettuani tein CSS gridillä graafikon antamaa layoutin mukailevan pohjan ja päivittelin uusia värejä ja fontteja sivustolle.

Onnistuin päivän tavoitteessa hyvin, sain päivitettyä sivuston ulkoasua huomattavasti modernimpaan suuntaan ja luotua hyvän pohjan mitä on helpompi muokata myös tulevaisuudessa.

Keskiviikko 14.02.2018

Asiakkaalta tuli tilaus lisäosalle Optiwise sovelluksen dashboard näkymään, mitä olen aikaisemmilla viikoilla jo tehnyt. Toiveissa oli saada yksinkertainen koontinäkymä, mikä näyttää yleisimmät kulutukseen liittyvät mittarit kaikissa liikkeissä samassa näkymässä. Koontinäkymää pitää pystyä myös lajittelemaan, kulutuksen, nimen, hinnan jne. Perusteella, näkymän tiedot pitää pystyä myös lataamaan csv formaatissa. Tästä päivän tavoitteeksi asettuikin luoda pohja ja raakaversio moduulissa näkyvästä datasta.

Sain päivän aikana moduulin raakaversion luotua, mistä oli hyvä lähteä seuraavana päivänä kehittämään valmista versiota. Sain myös loppupäivästä graafikoltamme layoutin, miltä viimeistellyn version tulisi näyttää.

Torstai 15.02.2018

Päivän tavoitteena oli saada eilen aloitettu koontinäkymä suurimmaksi osaksi valmiiksi.

Ensimmäiseksi päivitin tekemäni raakaversion vastaamaan graafikkomme näkemystä siitä, miltä koontinäkymän tulisi näyttää. Tämän jälkeen aloin lisäämään siihen isompia toiminnallisuuksia, mikä tuottikin omia haasteitaan. Kaikki koontinäkymässä tarvittava tieto oli jo saatavilla olemassa olevista rajapinnoistamme, mutta huomasin ensimmäistä versiota tehdessäni, että koodi näytti todella vaikeasti luettavalta eikä se noudattanut DRY periaatetta kovin hyvin. Ensimmäisen versioni ongelmat olivat siinä, että tein 9 eri REST kutsua samaan aikaan ja koska koontinäkymässä näytettävä data koostui kaikkien näiden kutsujen yhdistämisestä, sisälsi toteutukseni kovin pitkiä callbackejä, sekä promise.all() funktion käyttöä, mikä teki koodista kovin vaikealukuista. Koodin vaikealukuisuudesta johtuen aloin tutkimaan, miten sitä saisi siistittyä. Hetken googletuksen jälkeen löysinkin observableihin kuuluvan funktion nimeltä forkJoin(), millä voidaan kutsua montaa eri rajapintaa kerralla ja kun kaikki kutsut ovat onnistuneet, palauttaa niiden tulokset. Tai jos jokin kutsuista epäonnistuu niin virheilmoitus. forkJoinia käyttämällä pystyin ensinnäkin abstraktoimaan paljon toiminnallisuutta pois angularin komponentista ja siirtämään sen serviceen, sekä ylipäätään vähentämään paljon käytettyjä koodi rivejä ja tekemään koodista paljon luettavampaa.

Olin tyytyväinen aikaansaannoksiini päivällä, vaikka en saanutkaan komponenttia vielä täysin valmiiksi. Opin uuden erittäin käytännöllisen funktion olemassaolosta ja sain kirjoitettua ainakin omasta mielestäni siistiä koodia.

Perjantai 15.02.2018

Päivän tavoitteina oli viimeistellä sekä ZeeCloud portaalin ulkoasupäivitystä, sekä Optiwise sovellukseen tilattua koontinäkymää.

Aloitin päivän ZeeCloudin parissa, mikä olikin enää lähinnä pienten graafisten viilausten tekemistä, mikä sujuikin nopeasti. Tämän jälkeen pureuduin koontinäkymään ja aloin toteuttamaan siihen vielä tilattua csv:n tuonti ominaisuutta. Tätä logiikkaa ei onnekseni itse

tarvinnut toteuttaa, vaan siihen löytyi Papa Parse niminen kirjasto hetken googlettelun jälkeen. Papa Parse on toteutettu Javascriptiä varten, mutta siitä löytyi myös angularia varten portattu versio ngx-papaparse ja sen käyttö olikin suhteellisen yksinkertaista. npm:llä asennettiin vain paketti, minkä jälkeen dashboard moduulissa importattiin kyseinen kirjasto. Csv tiedoston luomiseksi käytettiin Papa Parsen unparse() funktiota. Kyseinen funktio ottaa helpoimmillaan parametriksi taulukon ja ottaa taulukon ensimmäisestä kohdasta sarakkeiden otsikot. Tämä aiheuttikin minulle pientä päänvaivaa, koska taulukossani olleet otsikot olivat englanniksi, ohjelmointikäytännöistämme johtuen. Sarakkeiden otsikot olivat myös väärässä järjestyksessä. Joten tein funktion, jota kutsutaan ennen csv tiedoston luontia, mikä kääntää taulukossa olevat nimet suomeksi ja asettelee myöskin sarakkeet oikeaan järjestykseen. Päivän lopuksi ilmestyi vielä yksi yllättävä bugi raportti liittyen erään komponentin graafi näkymään. Graafin alla on tarkoitus näkyä päivämääriä ja kellon aikoja, mutta jostain syystä kaikki ajat näkyivät aikana 00:00:00. Kyseinen ongelma on luultavasti ilmestynyt sen takia, kun päivitimme pari viikkoa sitten angularin 4.0 versiosta versioon 5.2.2. Vasta nyt joku oli huomannut virheen.

Onnistuin päivän tavoitteissa omasta mielestäni melko hyvin. En toki kerennyt loppupäivästä ilmennyttä bugia vielä korjaamaan, mutta se jääköön seuraavan viikon tehtäväksi.

Viikkoanalyysi

Viikko oli suhteellisen mielenkiintoinen, sisältäen minulle paljon uutta asiaa. En ollut aiemmin päässyt työskentelemään ZeeCloud pilviportaalimme kanssa, enkä myöskään aiemmin ollut koskenut djangoon ylipäätänsä.

Oli myös erittäin hienoa saada siistittyä omaa koodia, sekä löytää uusi hyödyllinen funktio forkJoin operaattori toimii erittäin hyvin tilanteissa, missä on monta observablea ja välität ainoastaan niiden lopputuloksista. Tästä on ainakin kaksi hyötyä, se selkeyttää koodia, koska ei tarvitse ketjuttaa useita callbackejä. Koodista tulee täten myös paljon helpommin luettavaa. forkJoin oli myös kätevä omassa käytössäni tässä tilanteessa, koska jos yksikin observable kohtaa virhetilanteen, ei palauteta minkään näiden tuloksia, joten tuloksien oikeellisuus pystytään näin ollen suhteellisen hyvin varmistamaan. (Learn RxJS 2018.)

3.3 Seurantaviikko 3

Maanantai 19.02.2018

Maanantain tavoitteena oli korjata edellisenä perjantaina löytynyt bugi, joka sai dygraphs kirjaston graafien alla olevat ajat näkymään aina muodossa 00:00:00. Jos bugiin löytyisi ratkaisu voisin loppupäivästä yrittää vielä viilata ZeeCloud pilviportaalimme ulkoasupäivitystä.

Aamusta aloin selvittämään, mistä ongelma dygraphs kirjaston kanssa johtuisi ja mistä se on mahtanut tulla. Kuten jo edellisenä perjantaina spekuloin, että vika olisi ilmestynyt angular kirjasto päivityksestä johtuen, niin en kovin pielessä arvauksineni ollut. Angular päivityksen yhteydessä tuli nimittäin päivitettyä kaikki muutkin päivitettävissä olevat kirjastot, joista yksi oli juurikin dygraphs. Meillä olikin ollut dygraph kirjastosta erittäin vanha versio käytössä mikä selvisi, kun selasi dygraphs projektin git historiaa pitemmälle. N. kaksi vuotta sitten oli jo kirjastoon tehty päivitys mikä liittyi käsillä olevaan ongelmaan. Käyttämämme funktiot olivat muotoa Dygraphs.DAILY, Dygraphs.SIX_HOURLY jne. Mutta kyseinen funktio oli jo kaksi vuotta sitten päivitetty muotoon Dygraphs.Granularity.Daily. Tämän jälkeen päivitin vain käyttämämme funktiokutsut oikeaan muotoon ja siirsin projektin testiympäristöömme (STAGING). Testiympäristössämme käymme aina testaamassa uusia muutoksiamme. Tarkoituksena selvittää hajosiko mikään muutosta tehdessä. Käytämme dygraphsiin pohjautuvia graafeja kahdessa eri paikassa sovelluksessamme. Samalla kun korjasin ilmentynyttä päivämäärä ongelmaa olin myös päivittänyt hiukan kyseisen komponentin ulkoasua, mikä näyttikin hyvältä, sillä sivulla missä päivämäärä ongelmakin ensin ilmeni. Olin vain ikävikseni unohtanut, että käytämme komponenttia kahdessa eri paikassa ja en ollut edes vilkaissut niistä toiseen (kartta komponentti). Kartta komponentissa graafi olikin täysin hajonnut. Joten loppupäivä menikin sen ulkoasua korjatessa.

Onnistuin puolittain päivän tavoitteissani, sain korjattua bugin mikä olikin pääprioriteettini, mutta onnistuin samalla aiheuttamaan uuden, mistä johtuen en kyennyt tekemään viimeisiä viilauksia ZeeCloud pilviportaalimme ulkoasuun, emmekä saaneet vielä korjauspäivitystämme siirrettyä tuotantoon, kartta komponenttiin ilmentyneen ulkoasun hajoamisen johdosta.

Tiistai 20.02.2018

Tiistain tavoitteena oli korjata maanantaina ilmestynyt ulkoasun hajoaminen kartta komponentissa ja viimein siirtyä ZeeCloud pilviportaalin ulkoasun viilaamiseen.

Jatkoin siitä mihin edellisenä päivänä jäin, eli kirjoittamaan uusiksi kartta komponentin html ja CSS tiedostoja. Ongelmaksi muodostui se, että karttakomponentissa dygraphin aikavälit miltä dataa haetaan, pystyy valita vain tasoilla päivä, viikko tai kuukausi. Aikaisemmin korjaamassani komponentissa pystyy itse asettamaan haluamansa aikavälin. Tästä johtuen en saanut kartta komponentissa heti testattua, että toimiiko muutokseni ulkoasuun vai ei, koska testidatani oli sen verran vanhaa, että kyseiset aikaväli valinnat eivät siihen riittäneet. Onneksi kollegani oli juuri aamulla saanut valmiiksi testidata generaattorin mikä sopikin tähän tilanteeseen paremmin kuin hyvin. Otin käyttööni backendistä git branchin, mikä sisälsi testidata generaattorin ja kollegani opasti minua sen käytössä. Generaattorin avulla sain omaan kehitysympäristööni tarvittavalle aikavälille generoitua dataa ja pystyin testaamaan miltä kartta komponentti näyttää. Komponentti alkoi näyttää hyvältä, joten oli aika siirtää se testiympäristöön ja käydä vielä läpi aiheuttiko se mitään ongelmia. Komponentti toimi ja näytti hyvältä myöskin testiympäristössä, joten tein muutoksesta merge requestin master branchiin, jonka jälkeen siirsimme muutokset tuotantoon. Tuotantoon siirron jälkeen piti käydä vielä kaikki instanssit läpi ja katsoa, että muutokset varmasti toimivat, eikä tarvitse rollbackata muutosta. Kaikki toimi niin kuin pitääkin ja muutos saatiin onnellisesti valmiiksi.

Onnistuin päivän päätavoitteessa, mutta ZeeCloudin ulkoasun viilaaminen antoi vielä odottuttaa itseään.

Keskiviikko 21.02.2018

Deadlineksi tälle päivälle tarkoituksena oli ottaa lokaalissa testiympäristössäni käyttöön uusi moduuli erp järjestelmäämme ja testata sen toimivuutta.

Päivä alkoi sillä, että käynnistelin lokaalin asennuksen erp järjestelmästämme (Odoo). Asensin siihen Online Proposals moduulin, millä on tarkoitus tehdä tarjouksia asiakkaille mitkä he voivat kuitata online lomakkeella. Kerkesin juuri ja juuri asentamaan moduulin ja katsomaan miltä se näyttää vakio asetuksillaan, kun sainkin tietää, että minun pitäisi siirtyä viimeistelemään ZeeCloud pilviportaalimme ulkoasu, että se saataisiin sisäisesti vielä arvioitavaksi tällä viikolla.

Aloin siis käymään läpi graafikolta saamaani layoutia ja vertasin sitä jo tehtyihin muutoksiin ja vaikutti siltä, että jäljellä ei ollut juuri muuta, kuin fonttien muuttelua. Nopeasti kävi kuitenkin selväksi, että aiemmin tekemissäni muutoksissa oli suuremman luokan ongelma. Ongelma oli aikaisemmin tekemässä CSS grid pohjassa. Olin tehnyt pohjaan vasemmassa sivussa olevan linkkejä sisältävän valikon, mikä itsessään toimi kyllä ihan hyvin. Ongelmaksi muodostui se, että joillakin sivuilla saattoi olla niin paljon sisältöä, että pystyäkseen painamaan sivupalkin alalaidassa olevaa nappia, joutui sivua rullata alaspäin kohtuuttoman paljon. En ollut huomannut tätä lokaalissa kehitysympäristössäni, koska minulla ei ollut siinä juuri yhtään dataa, joten ongelma ilmeni vasta, kun katsoin tekemiäni muutoksia staging ympäristössämme. Ei myöskään ollut soveltuvaa, että nappi saattoi sijaita pitkän rullauksen takana, koska sitä painamalla pystyi pienentämään sivupalkin näyttämään vain ikoneja tekstien sijaan. Tämän takia siitä myös muodostui pykälää suurempi ongelma, jos nappi sijaitsisi aina samassa kohdassa sivupalkissa, sen olisi helposti pystynyt korjaamaan asettamalla pelkälle napille ominaisuudet *position: fixed, bottom: 1rem, left: 1rem,* mutta näin ei voinut tehdä, koska nappi sijaitsi sivupalkissa, minkä koko muuttui, kun nappia painoi ja koon muuttuessa napin sijainnin piti myös muuttua. Tästä johtuen jouduinkin hieman selvittämään, miten saisin koko sivupalkin toimimaan position: fixedinä, mutta sekään ei ollut ihan yksinkertaista, sillä CSS elementtiä ei voi muuttaa fixed positioon, jos se sijaitsee CSS gridin sisällä, koska silloin kyseinen elementti poistuu grid asettelusta. Tein tätä varten nopeasti uuden pikku projektin, missä testailin, miten saisin sivupalkin/napin toimimaan haluamallani tavalla. Lopulta ratkaisuksi muovautui sellainen, että sivu ei voinut olla yhden CSS gridin sisällä vaan se piti jakaa kahteen erilliseen gridiin, joista toinen on sivupalkki ja toinen itse sivuston sisältö. Sillä vaikka gridin sisäisiä elementtejä ei voi muuttaa positioon fixed, hajottamatta itse gridiä. Koko grid kyllä voi olla positiossa fixed. Sain kyseisen ominaisuuden toimimaan pikaisesti tehdyssä pikkuprojektissani ja seuraavalle päivälle jäikin sitten kyseisen toiminnallisuuden siirtäminen itse ZeeCloud projektiin.

Päivän tavoitteet muuttuivat täysin päivän edetessä ja niitä ei tullut saavutettua. Opin kuitenkin uutta liittyen CSS gridiin, mistä olen oikein tyytyväinen.

Torstai 22.02.2018

Päivän tavoitteena oli saattaa eilen aloitettu ZeeCloud pilviportaalin sivupalkin korjaus maaliin.

Aloin selvittämään miten saisin toteutettua eiliseen mini projektiini keksimän prototyyppi ratkaisun itse ZeeCloud projektissa. Aluksi tulin siihen tulokseen, että ainoa vaihtoehtoni

on kirjoittaa koko sivuston CSS uusiksi. Päädyin tähän ratkaisuun siksi, koska CSS:ää ja html:ää oli jo kirjoitettuna erittäin paljon ja tuntui lähes mahdottomalta selvittää, miten sen saisi toimimaan haluamallani tavalla. Aloitin siis sivuston CSS:n ja html:n kirjoittamisen puhtaalta pöydältä ja uhrasin tähän jonkin verran aikaakin, kunnes tajusin, että kaiken uudelleenkirjoittamisessakaan ei olisi kyllä mitään järkeä. Palautin CSS:n ja html:n alkuperäiseen tilaan git checkouttaamalla kyseiset tiedostot. Keksin, että muutoksen toteuttamiseksi riittäisi, että nykyinen CSS gridi, muutettaisiin sisältämään vain sivupalkki, koska sitä varten on kirjoitettu eniten CSS:ää. Sivustojen itse sisällöt olivat hyvin yksinkertaisella pohjalla, joten irrotin ne gridistä ja tein niille oman gridin. Tämän jälkeen muutin sivupalkin gridin positioon fixed, ja sivupalkki vaikutti toimivan juuri niin kuin pitääkin. Tämän jälkeen vielä siirsin kehitysversioni staging ympäristöön, selvittääkseni ilmeneekö mitään eriskummallisuuksia isommalla datamäärällä. Isolla datamäärällä sivupalkkia pienentäessä sivuston sisältö siirtyi ikään kuin yhden sivun verran alaspäin, mikä jääkin sitten seuraavan päivän ongelmaksi.

Onnistuin itse päivän tavoitteessani eli sivupalkin korjaamisessa hyvin, mutta onnistuin samalla luomaan itselleni uuden bugin korjattavaksi.

Perjantai 23.02.2018

Päivän tavoitteena oli korjata ZeeCloud pilviportaalin sisällön alaspäin siirtyminen, kun sivupalkin pienentää.

Hetken aikaa tutkiskeltuani tulin siihen tulokseen, että sisältö ei osannut jostain syystä pysyä ylhäällä, kun palkin pienensi. Oletan, että tämä johtui siitä, että täyskokoisena palkki on 200px leveä ja sivuston sisältö on siten asetettu margin-left: 200px:llä, mutta palkin kun pienentää. Se on 65px leveä ja samalla vaikka Javascriptillä muuttaa margin-left asetuksen 65px, ei sivu osaa asettua sille annettuun uuteen tilaan. Ratkaisuksi tähän muutin myöskin sivuston sisällön muotoon position:fixed, top:0, silloin kun sivupalkki on pienennettynä. Tämä ratkaisu vaikutti toimivan ongelmitta. Esittelin vielä kokonaisuuden muutokset pyytäneelle esimiehelle. Muuten muutokset saivat hyvää palautetta, mutta kahdelle sivulle piti laittaa vielä uudistettu logo/taustakuva. Uusi logo ja taustakuva eivät olleet aikaisemmin tulleet puheeksi, joten niitä ei ollut olemassa ja niiden tekeminen jäi graafikomme vastuulle.

Päivän tavoitteet tulivat täytettyä hyvin, enkä onnistunut luomaan uusia ongelmia, joten kaiken kaikkiaan hyvä päivä.

Viikkoanalyysi

Tämän viikon teemana oli virheiden etsiminen ja korjaaminen, sekä erinäisten CSS muutoksien toteuttaminen. Asia mikä tuli huomattua on se, että käytössä olevat kirjastot kannattaa yrittää pitää ajan tasalla ja niissä oleviin mahdollisesti koodia muuttaviin muutoksiin kannattaa kiinnittää huomiota. Varsinkin kolmannen osapuolen kirjastoissa on usein ongelmia siinä, kuinka ne pysyvät kehityksessä mukana. Angularin kanssa vielä erityisiä ongelmia aiheuttaa se, että itse framework kehittyy kovaa tahtia ja isot versiopäivitykset saattavat aiheuttaa yhteensopivuus ongelmia käytössä olevien kirjastojen kanssa. Huonolla tuurilla jotain kolmannen osapuolen kirjastoa on kehittänyt vain yksi tai muutama tekijä ja heillä ei ole resursseja pysyä nopeassa kehityksessä mukana, tai kehitys saattaa jopa yllättäen loppua. Kirjastoja valitessa onkin hyvä ottaa huomioon, onko kyseisellä kirjastolla useampia kehittäjiä, kuinka usein siihen on tehty päivityksiä ja myös latauksien määrä voi antaa osviittaa kirjaston toimivuudesta.

Sain viikon aikana myös jälleen kehitettyä osaamistani CSS gridin parissa. Nautin CSS gridistä, erittäin paljon, vaikka toki sen käyttäminen vanhassa ympäristössä ei ole niin mutkatonta, kuin uudessa. Sitä käyttäessäni en ole minkään tietyn kirjaston varassa ja pystyn helposti muotoilemaan sivut halutun laiseksi ja myös tarvittaessa skaalaamaan ne nopeasti mobiiliympäristöön sopivaksi.

CSS grid layout on tehokkain layoutjärjestelmä saatavilla CSS:n parissa. Se on kaksiulotteinen järjestelmä, mikä tarkoittaa, että se pystyy käsittelemään niin sarakkeita, kuin rivejäkin, toisin kuin flexbox, mikä on pääosin yksiulotteinen järjestelmä. Gridin kanssa työskennellään pääasiassa kohdistamalla CSS sääntöjä niin vanhempi elementtiin, kuin sen lapsiinkin. (House 2018.)

3.4 Seurantaviikko 4

Maanantai 26.02.2018

Päivän agendana yrittää sovittaa ZeeCloudissa olevia iframeja taustaan ja erilaisten mouseover hoverien lisääminen muutamaan li elementtiin.

Päivä alkoi nopealla viikkopalaverilla, missä käytiin läpi mitä kenelläkin on työn alla tällä hetkellä. Selvitettiin myös mitä työt mitä on tulossa työn alle ja onko mitään ongelmia missä tarvitsisi apua tms.

ZeeCloud koostuu useasta eri widgetistä, eli pienoissovelluksesta. Nämä widgetit on toteutettu iframeilla ja niillä ei ole mitään standardi kokoa. Tästä johtuen osa istuu hyvin ja osa vähän vähemmän hyvin sivulle. Suurimman iframen ongelmana oli, että se meni yli sivun leveyden ja aiheutti turhaa sivuttaissuunnan rullaamisen tarvetta. Ratkaisin tämän rajoittamalla iframen maksimi leveydeksi 90vw, millä iframe pakotettiin pienemmäksi, kuin se oli widgetissä asetettuna. Tästä johtuen widgetti oli aavistuksen liian kapea ja sivuston oma tausta erosi iframen taustasta häiritsevän paljon. Kaikilla iframeilla ei kuitenkaan ole samanvärinen tausta, joten jouduin asettamaan taustaväriksi geneerisen harmaan, joka ei pistä liian pahasti silmään minkään iframen kohdalla. Muutamassa kohdassa sivulla teksti oli valkoista ja viemällä hiiren sen päälle muuttui myös tausta valkoiseksi, eikä tekstiä näin ollen pystynyt lukemaan. Muutin siis hover taustavärin graafikon määrittelemäksi oranssin sävyksi.

Päivä oli suhteellisen helppo ja sain tehtyä kaiken mitä oli tarkoituskin.

Tiistai 27.02.2018

Tiistain tavoitteena oli tutkia erp järjestelmäämme (odoo) saatavilla olevaa online proposals moduulia, testata sen toimintaa kunnolla lokaalisti ja selvittää soveltuuko se käyttöömme.

Olin viime keskiviikkona kerennyt asentamaan online proposals moduulin lokaaliin ympäristööni ja nyt oli aika testata sitä vähän kattavammin. Moduulin tarkoitus yrityksemme käytössä olisi virtaviivaistaa uusien asiakas projektien luomista ja vanhoihin jatkuviin projektiin tehtävien muutoksien hyväksyttämistä. Ideana siis lähettää asiakkaan sähköpostiin erp järjestelmämme kautta linkki, mistä asiakas voi avata ”tarjouksen” ja siihen sisältyvät tiedot, kuten vaikka tuntimäärän ja hinnan. Kun asiakas hyväksyisi tämän, merkintä siitä ilmestyisi välittömästi erp järjestelmäämme ja asiaan liittyvät henkilöt voisivat alkaa työskennellä ominaisuuden / projektin parissa.

Pienten konfigurointi selvitysten jälkeen sain moduulin lähettämään sähköposteja itselleni ja pystyin katsastamaan ihan oikeaa toiminnallisuutta. Moduuli vaikutti erittäin pätevältä, kun tehdään uusia projekteja, koska sen sai automaattisesti luomaan uuden projektin erppiin, kun asiakas oli sen käynyt hyväksymässä. Ongelmaksi taas ilmeni se, että jo olemassa oleviin projekteihin online proposalsin kautta lisätyt uudet työt eivät toimineet, kuten oletettiin. Ideana oli, että uudet taskin luominen vanhaan projektiin ei automaattisesti näyttäisi laskutuksessa kaikkia kyseisessä projektissa olevia tunteja, mutta ikävä kyllä se

niin teki. Tämä taas aiheuttaa paljon ylimääräistä työtä asianosaisille, kun pitää käydä manuaalisesti muokkaamassa laskutusten tunteja.

Päivän lopuksi kirjoitin raportin huomioistani liittyen moduuliin ja lähetin sen eteenpäin. Päivän tavoitteena oli online proposals moduuliin tutustuminen ja sen sain suoritettua ongelmitta.

Keskiviikko 28.02.2018

Graafikkomme oli saanut eilen valmiiksi uudet logot ZeeCloudiin, joten päivän tavoitteina oli laittaa ne paikoilleen.

Aloin tutkimaan graafikon antamaa uutta layoutia, siitä mihin paikkoihin uudet logot ja kuvat tulisi sijoittaa. Kirjautumissivun kirjautumisruudun yläpuolelle oli tarkoitus asettaa keskitetysti uusi logo. Vanha kirjautumissivu oli toteutettu vähän vaikeasti käyttämällä erilaisia CSS määritelmiä, kuten position: absolute, kovakoodattuja margineita jne. Tästä johtuen logon lisääminen kirjautumisruudun päälle olisi ollut aika ikävää, sekä rumaa koodia. Päätin siis kirjoittaa koko kirjautumissivun uusiksi käyttämällä CSS gridiä. Tarkoituksena keskittää kirjautumisruutu keskelle sivua ja logo sen päälle keskelle. Sivun uudelleen kirjoitus meni rutiinilla ja logo istui kirjautumisruudun päälle täydellisesti. Tämän jälkeen vaihdoin vielä yhtä taustakuvaa, kun käyttäjä luo itselleen uuden dashboardin, missä ei ole vielä sisältöä.

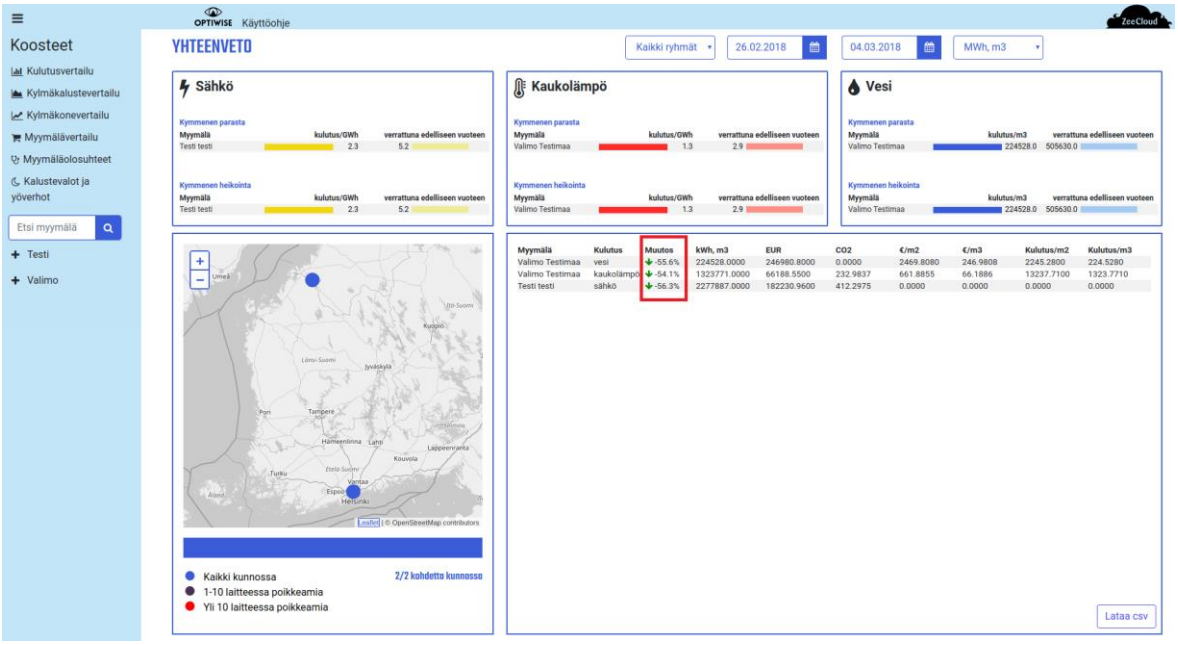
Päivän tavoitteet onnistuivat ja mitään ei jäänyt tekemättä.

Torstai 01.03.2018

Torstain tavoitteena oli saada ZeeCloudin pienennetty sivupalkki toimimaan ja lisätä Optiwise dashboardin taulukko näkymään uusi sarake.

Aloitin päivän selvittämällä, miten saisin pienennetyn sivupalkin muistamaan aiemmin painetun alilinkin. Aluksi ajattelin toteuttaa kyseisen toiminnallisuuden jQueryllä, mutta se vaikutti suhteellisen työläältä noin yksinkertaiseen asiaan. Päätin siis vähän lueskella djangon dokumentaatiota ja sain selville, miten djangon templateissa pystyy käyttämään if else lauseita, mikä yksinkertaisti tehtävääni suuresti ja niitä käyttämällä sain sivupalkin toimimaan halutulla tavalla.

Asiakkaalta oli tullut pyyntö lisätä optiwise dashboardin taulukko näkymään uusi sarake (kuvio 3), mistä näkisi nopealla silmäyksellä, kuinka paljon eri kulutukset ovat muuttuneet prosenteissa verrattuna edelliseen vuoteen. Lisäksi, kun latasi tiedot csv:nä tuli muutosten erotus näkyä myös lukuina. Toiminnallisuuden toteuttamisessa itsessään ei ollut mitään ongelmia, se toki vaati kolme uutta kutsua REST-rajapintaan. Nämä kutsut sain kätevästi lisättyä jo aiemmin toteuttamaani funktioon mikä käytti observable.forkJoin() funktiota.



Kuvio 3. Dashboard komponentti (lisätty sarake rajattu punaisella)

Onnistuin päivän tavoitteissani mallikkaasti, eikä mitään jäänyt tekemättä.

Perjantai 02.03.2018

Päivän tavoitteena oli niinkin yksinkertaisen asian, kuin linkin luominen dashboardin kulutus koosteesta itse kulutusvertailukomponenttiin.

Vaikka linkin toteuttaminen kuulostaakin yksinkertaiselta, mutta linkin mukana pitäisi vielä siirtyä parametrit mitä dashboardilla on käytetty. Parametrien siirto itsessään ei myöskään ole mikään vaikea tehtävä, mutta kulutusvertailu komponentti on erittäin suuri ja monimutkaisesti toteutettu kokonaisuus, joten itse komponentin alustaminen kyseisillä parametreillä aiheutti suurta päänvaivaa. Sain kyllä valitut päivämäärät toimimaan tarkoituksenmukaisesti komponenttia alustettaessa, mutta valittujen sivustojen id:t ja niiden perusteella komponentin alustaminen tuotti suunnattomia vaikeuksia.

En onnistunut päivän tavoitteessani saada linkkiä dashboardista kulutusvertailu komponenttiin. Kulutusvertailukomponentti on toista tuhatta riviä pitkä ja joudun sen kanssa kyllä painimaan vielä lisää ensi viikolla.

Viikkoanalyysi

Sain tehtyä onnistuneita muutoksia ZeeCloudiin, sekä testattua online proposals moduulia Odoossa. Sain myös lisättyä uuden sarakkeen dashboard moduulin taulukkonäkymään. Silti viikosta jäi päällimmäisenä mieleen kulutusvertailu komponentin alustamiseen liittyvät ongelmat ja tämä kyllä jäi kaivelemaan minua.

Odoo on siitä kätevä ERP ohjelmisto, että se on helposti laajennettavissa moduuleilla ja siitä löytyy ilmainen versio, mikä meilläkin on käytössä. Siihen löytyy paljon erilaisia ohjelmia ja moduuleita millä sen toiminnallisuutta pystyy laajentamaan vastaamaan yrityksen tarpeita. Online proposals moduulin testaaminen onnistui omasta mielestäni hyvin ja ilman suurempia ongelmia.

Kulutusvertailukomponentti aiheutti kyllä suurta päänvaivaa ja tässä vähän kostautuu se, että varsinkaan kyseisen komponentin kohdalla ei ole noudatettu angularin virallista tyyliopasta. Tyylioppaassa (Angular 2018.) mainitaan heti ensimmäisessä kohdassa, että on suositeltavaa rajoittaa jokaisen yksittäisen tiedoston, kuten palvelun tai komponentin koko 400:n riviin koodia. Tähän suositukseen on näin jälkiviisaana helppo yhtyä, sillä kulutusvertailu komponentti on erittäin vaikea lukuinen ollessaan toista tuhatta koodi riviä pitkä komponentti. Komponentissa on vielä niin monia eri toiminnallisuuksia, mitkä olisi pitänyt pilkkoa pienempiin osiin, jotta sen läpikäyminen olisi helpompaa.

3.5 Seurantaviikko 5

Maanantai 05.03.2018

Päivän tavoitteina oli saada linkki toimimaan parametreilla kulutusvertailukomponentissa, sekä laittaa viimeviikolla tehdyt muutokset Optiwise sovelluksesta staging ympäristöön ja ZeeCloud sovellus erilliseen beta ympäristöön.

Aamu alkoi sillä, että tein ZeeCloud ja Optiwise sovelluksista git merge requestit asiaankuuluviin haaroihinsa. Kollegani, joka kävi merge requesteja läpi, antoi palautetta, että voisin lisätä kommentteja Optiwise sovellukseen tekemiini muutoksiin ja kuvailla, mitä eri

funktiot siellä tekevät. Lisäsin kommentit ja puskin muutokset gittiin, näillä muutoksilla merge requestini hyväksyttiin.

Sitten pääsinkin päivän pähkinään, eli kulutusvertailukomponentin lataukseen parametreilla. Kyseessä on erittäin suuri komponentti, missä on pelkkiä alustusparametreja n. 100 riviä. Aikani yritin päästä selville komponentin sielunelämästä, mutta jouduin lopulta pyytämään kollegaltani apua sen kanssa. Kollegan kanssa vähän aikaa yhdessä katsottuamme totesimme, että on parempi tehdä uusi komponentti, mikä hallitsee eri filttereiden määrittelyä, mistä ne emitoidaan sitten kulutusvertailukomponenttiin. Teimme uuden komponentin mihin määrittelimme pohjan kaikista asioista mitä filtterin tulisi sisältää. Ja saimme sen jo osittain toimimaan kulutusvertailukomponentin kanssa.

Loppupäivä oli kohtuullisen kimurantti ja kulutusvertailukomponentin kanssa jäi vielä paljon tehtävää, mutta pääasia, että edistystä tuli ja oli kiva saada kollegalta näkemystä, miten ongelma tulisi ratkaista.

Tiistai 06.03.2018

Päivän tavoitteena oli jatkaa kulutusvertailukomponentin uudelleenkirjoitusta ja sen kaveriksi tehdyn filtteri komponentin kirjoittamista, jotta saisin kulutusvertailukomponentin toimimaan, niin parametreilla, kuin ilmankin.

Jatkoin siis siitä mihin eilen jäin. Kirjoittelin muutamia change event listenereitä ja niiden toiminnallisuuksia filtteri komponenttiin, niiden ideana oli siis lähettää dataa filtteri komponentista takaisin kulutusvertailu komponenttiin. Ideana se, että aina muutoksen tapahtuessa lähetetään päivitetty filtteri kulutusvertailukomponenttiin ja päivitetään näkymä vastaamaan filtterin antamia tietoja. Jouduin myös päivittelemään event listeneriä itse kulutusvertailukomponentissa ja muokkaamaan miten se reagoi tiettyjen filtterimuutosten päivityksiin.

Onnistuin päivän tavoitteissa kohtalaisesti. Olisin halunnut saada enemmänkin aikaan, mutta tiettyjen kulutusvertailukomponentin läpi selaaminen ja tutkiminen vie paljon aikaa, että oikeasti ymmärtää mitä missäkin tapahtuu ja pystyy sitten muokkaamaan sen toimintaa.

Keskiviikko 07.03.2018

Päivän agendana jatkaa yhä kulutusvertailukomponentin ja siihen liittyvän filtteri komponentin kirjoitusta/uudelleenkirjoitusta.

Aloitin päivän jatkamalla siitä mihin eilen jäin, eli event listenerien toiminnallisuuksien kirjoittamiseen. Nopeasti kuitenkin kohtasin vanhan ongelmani, eli sen, että en vieläkään saanut oikean sivun datoja ladattua, kun tulin parametreillä kulutusvertailusivulle, vaan komponentti latasi kaikkien mahdollisten sivujen datat. Tähän kysyin kollegaltani apua ja hetken aikaa yhdessä tutkittuamme hän keksi hyvinkin yksinkertaisen ratkaisun ongelmaani. Piti vain asettaa boolean muuttuja, minkä perusteella dataa filtteröitiin riippuen siitä tultiinko sivulle parametreilla vai ilman. Tämän jälkeen ongelmaksi muodostui, että sivulle tultaessa komponentti lähetti toistakymmentä kutsua backendiin, vaikka sen pitäisi lähettää niitä tilanteesta riippuen, enimmillään kuusi. Tähän tilapäiseksi ratkaisuksi asettui setTimeout funktion käyttö, millä annettiin komponentille aikaa ladata sen child komponentit valmiiksi, mikä esti liian monen kutsun lähettämisen. Tämä pitänee myöhemmin päivittää käyttämään Rx/js observable debouncea, mutta nyt oli vain tärkeää saada komponetti alustavasti toimimaan.

Tässä välissä kollegani teki merge requestin develop haaraan. Tarkistin hänen tekemät muutokset ja annoin pari pientä korjauspyyntöä liittyen unohtuneisiin kommentteihin ja konsoli logituksiin. Hän korjasi huomautukset, minkä jälkeen hyväksyin merge requestin.

Tämän jälkeen palasin taas kulutusvertailu komponentin pariin. Siinä on käytössä ns. breadcrumb, mikä päivittyy riippuen zoomi tasosta. Sen avulla käyttäjä pystyy seuraamaan, minkä päivämäärän tietoja milloinkin selataan ja palaamaan takaisin aiempiin valintoihin. Filtteri komponentin käyttöönoton takia, tämä toiminallisuus pitää kirjoittaa uudelleen, koska tässä tapauksessa pääkomponentin pitää lähettää dataa lapsikomponenttiin, mikä tapahtuu @ViewChild dekoraattoria käyttämällä. Pääsin alkuun tämän toiminnallisuuden kirjoittamisessa ja siitä onkin hyvä jatkaa huomenna.

Sain edistystä aikaiseksi kulutusvertailu ja filtteri komponentin parissa, mutta olisin vieläkin halunnut saada enemmän aikaiseksi. Positiivisena kaneettina se, että olen pikkuhiljaa oppinut kysymään enemmän asioita, ilman että väkisin hakkaan päätä seinään pitemmän aikaa.

Torstai 08.03.2018

Päivän tavoitteena oli jatkaa kulutusvertailu komponentin uudelleen kirjoitusta ja julkaista uusi versio optiwise sovelluksesta.

Aloitin päivän jatkamalla siitä mihin eilen jäin. Eli kirjoitin lisää toiminnallisuutta parent -> child komponentin tietojen välittämiseen. Edistyin komponentin uudelleenkirjoittamisessa ihan hyvin, mutta ominaisuuteen tehtäväksi tarkoitetut tunnit alkoivat täyttyä, joten jouduin toistaiseksi lopettamaan kulutusvertailu komponentin uudelleen kirjoituksen.

Oli siis aika julkaista uusi versio Optiwise sovelluksesta tuotantoon. Uudessa versiossa oli toistaiseksi tehdyt muutokset dashboardiin, ilman linkkiä kulutusvertailuun. Tein siis merge requestin develop branchista master branchiin. Tämän jälkeen siirsin master branchin jenkinssillä tuotantoon ja testasin nopeasti version toimivuuden. Todettuani sen toimivaksi päivitimme kaikki tuotannossa olevat imaget uuteen versioon ja kävimme vielä nopeasti jokaisen instanssin läpi virheiden varalta.

Harmillisesti en pystynyt jatkaa kulutusvertailukomponentin uudelleenkirjoitusta loppuun asti, toivottavasti sille vielä joku päivä löytyisi aikaa, koska käytössä oleva versio on koodin osalta melko epäselvä. Optiwisen uuden version saaminen tuotantoon kuitenkin onnistui hyvin ja niiltä osin olen tyytyväinen päivään.

Perjantai 09.03.2018

Päivän tavoitteena kirjoittaa filtteri kylmäkalustesivulla näkyviä koneoppimisen avulla hankittuja tietoja varten.

Aloitin siis lisäämällä uuden napin taulukkoon, mistä pystyisi checkboxien avulla valitsemaan, mihin kategoriaan kuuluvia tuloksia haluttiin nähdä. Toteutin valikon bootstrapin popoveriin, eli nappia painamalla ilmestyi napin vasemmalle puolelle laatikko kaikista valittavista vaihtoehdoista. Tämän jälkeen aloin tekemään itse toiminnallisuutta, jotta valintoja muuttamalla muuttuisi myös näytettävät tiedot. Tätä varten oli tehty uusi backend rajapinta, joten ensimmäiseksi jouduin tekemään uuden funktion, millä haettiin dataa uudesta rajapinnasta. Kyseistä funktiota oli tarkoitus kutsua aina, kun valittiin tai poistettiin valinta checkboxista. Tätä varten tein myös objektin, missä oli kaikki checkboxien arvot ja boolean muuttuja millä seurattiin objektin eri arvojen checked tilaa.

Saavutin omasta mielestäni päivän tavoitteen, sillä filtterin valintoja muuttamalla muuttui myös sivulla näkyvä data valintoja vastaavaksi. Kyseessä ei sinällään ollut mikään mullistava ominaisuus, mutta en tämän tyylistä filtteriä ollut kirjoittanut hetkeen, joten se oli mukavaa vaihtelua.

Viikkoanalyysi

Viikko kului pääasiassa kulutusvertailu komponentin uudelleenkirjoittamiseen, toki loppuviikosta tuli myöskin uusi ominaisuus tehtyä ml filtterin muodossa. Alan pikkuhiljaa ymmärtämään koodin kommentoinnin merkityksen. Juuri tekemäni koodi on yleensä tietysti itselleni selkeää ja ymmärrettävää, mutta se ei sitä välttämättä ole muille ihmisille, varsinkaan heille jotka eivät kyseisen koodin kanssa työskentele. Myöskin muuttujien, funktioiden ja luokkien niminä on järkevää käyttää jotain mistä voisi ymmärtää mikä niiden tarkoitus on.

Hyvien nimien keksiminen vie aikaa, mutta se säästää enemmän, kuin se ottaa. Joten ole tarkkana nimien kanssa ja vaihda niitä, kun parempia tulee kohdalle. Kaikki jotka lukevat koodiasi ovat iloisempia, kun teet niin. Muuttujan, funktion tai luokan nimen tulisi vastata kaikkiin isoihin kysymyksiin. Sen pitäisi kertoa miksi se on olemassa, mitä se tekee ja miten sitä käytetään. Jos nimi tarvitsee kommentin se ei paljasta tarkoitustaan. (Martin 2009, 18.)

Kannattaa myöskin yrittää välttää huonojen kommenttien kirjoittamista, koska niistä voi olla enemmän haittaa, kuin hyötyä. Itsestään selvyyksien kommentoimisesta ei ole mitään hyötyä kenellekään. Tämän takia, jos joudun kirjoittamaan kommentteja, vaikka monimutkaisempiin luokkiin.

3.6 Seurantaviikko 6

Maanantai 19.03.2018

Päivän tavoitteena oli tehdä asiakkaalta saadun palautteen perusteella erilaisia pieniä ulkonäöllisiä korjauksia Optiwise sovelluksen dashboard näkymään.

Aloitin päivän tekemällä erilaisia pieniä korjauksia dashboardin koonti ja taulukkonäkymiin. Korjaukset sisälsivät erilaisten numeroiden muotoilujen muuttamista. Tällä hetkellä numerot pyöristyivät aina kokoluokasta riippumatta neljän desimaalin tarkkuuteen, koska jotkut vertailtavista luvuista saattavat olla hyvin pieniä. Korjasin nämä tekemällä dashboard moduuliin pipen. Pipet ovat angularin tapa muuttaa näytettävä data haluttuun muotoon html:n

sisällä. Tein pipen, joka palautti arvot, jotka olivat suurempia tai yhtä suuria kuin yksi pyöristettynä lähimpään kokonaislukuun, Javascriptin toFixed(0) funktion avulla. Numerot mitkä olivat pienempiä kuin yksi palautin Javascriptin toPrecision(1) funktion avulla, mikä palauttaa luvun ensimmäiseen merkkaavaan desimaaliin asti.

Tämän jälkeen kollegani huomasikin ihan muita asioita tehdessään bugin optiwise sovelluksen raportti näkymässä. Aloin siis seuraavaksi korjaamaan kyseistä bugia. Raportti näkymässä on kalenteri, minkä avulla voit valita kaksi päivämäärää ja raporttiin tuleva data haetaan näiden päivämäärien välillä. Jostain syystä raportti kuitenkin näytti viikonpäivät OBOE:na (off-by-one error), mikä ilmeni molemmissa päissä valintoja, eli jos valitsi vaikka päivät maanantaista sunnuntaihin, näytti raportti päivät tiistaista maanantaihin. Alkupään virhe johtui Javascriptin natiivi date funktion käytöstä ja korjaantui lisäämällä -1 tämän funktion lopputulokseen. Loppupään bugia en saanut vielä selvitettyä ja se jää tiistai aamulle selvitettäväksi.

Päivän alkuperäiset tavoitteet olivat suhteellisen yksinkertaisia, eikä niiden tekemisessä ollut suurempia ongelmia. Oli kiva pitkästä aikaa käyttää angularin pipejä, ettei niiden käyttö aivan unohdu. OBOE bugi taas vaikutti loppupään aikavalinnan osalta vähän vaikeammalta selvittää ja siihen pitääkin pureutua syvällisemmin tiistaina.

Tiistai 20.03.2018

Päivän tavoitteina oli selvittää eilen ilmennyt OBOE bugi, sekä ottaa tuotantokäyttöön online proposals moduuli erp järjestelmässämme.

Aloitin päivän selvittelemällä OBOE bugia, joka siis oli kalenterin aikavalinnan loppupäähän ilmestynyt ”ylimääräinen” päivä. Hetken aikaa ongelmaa ihmeteltyäni katsoin, minkälainen kutsu backendiin lähtee. Lähtenyt kutsu tapahtui kyllä oikeilla päivämäärillä, joten katsoin chromen debug työkaluilla itse vastausta minkä sain backendistä. Huomasin, että backendistä tuli jostain syystä yhden ylimääräisen päivän datat ja pyysin kollegaani selvittämään asiaa. Kollega huomasi bugin backendissä, mikä aiheutti kyseisen ongelman. Hänen tehtyä vaadittavan korjauksen siirsin muutokset staging ympäristöömme ja testasin ja totesin, että muutokset toimivat niin kuin pitääkin.

Seuraavaksi asensin online proposals moduulin tuotantoympäristöömme ja selvittelin miten siinä mukana tulevan vakio etusivun saisi poistettua. Pienen lokaalissa testiympäristössä olevan selvittelyn jälkeen keksin, miten sen saa vaihdettua normaaliksi sisäänkirjautumissivuksi. Ongelmaksi kuitenkin jäi vielä se, että jos meni samalle sivulle uudestaan

sisäänkirjautumisen jälkeen, se pyysi käyttäjätunnuksia vaikka olit jo kirjautuneena. Tämä saattaisi hämmentää käyttäjiä, joten pyrin selvittämään miten sen saisi pois. Tämä vaatii syvällisempää tutustumista itse Odoon koodiin, mihin en ole aiemmin vielä hirveästi perehtynyt. Selvitystyö jatkunee keskiviikkona.

Sain selvitettyä optiwisen OBOE bugin ja sain otettua online proposals moduulin tuotantokäyttöön, joten niiltä osin päivän tavoitteet sujuivat hyvin. Itse Odoon syvällisempi muokkaus vaatii vielä opettelemista, mikä vienee aikansa.

Keskiviikko 21.03.2018

Tälle päivälle ei ollut mitään erityisempiä tavoitteita, lähinnä Odoo:n kehitykseen tutustumista.

Aloitin päivän hajottamalla lokaalissa ympäristössä olevan Odoo asennukseni niin, että en päässyt enää kirjautumaan siihen sisään. Aikani selviteltyä ongelmaa, tulin siihen tulokseen, että on parempi siirtää Odoon kehitysympäristö dockerin päälle. Tämä ihan vain siitä yksinkertaisesta syystä, että jos hajotan ympäristön, on se nopea vain tuhota ja pystyttää uudelleen. Odoo docker ympäristön pystytys oli myös melko helppoa, kiitos Odoon tarjoaman valmiin docker imagen. Tein tähän vielä simppelin docker.compose tiedoston, mikä mounttaa kustomoidut Odoo lisäosat dockerin käyttöön. Tämän jälkeen ajoin ympäristön käyttöön docker-compose up komennolla, eikä sen enempää vaadittu. Tämän jälkeen asentelin vaadittavat moduulit Odooseen, jotta se vastaisi tuotantoympäristöä.

Loppupäivänä sainkin tehtäväksi tehdä merge requestin aikaisemmilla viikoilla tekemistäni muutoksista Optiwise sovellukseen, minkä jälkeen vielä ajoin muutokset jenkinssillä testiympäristöömme ja testasin nopeasti, että kaikki toimii.

Olin itseasiassa tyytyväinen päivääni, vaikka se ei nyt mitään akuuttia tehtävää sisältänytkään. Opin vähän enemmän Odoon kehitysympäristöstä ja oli mukavaa saada sekin pyörimään dockerin päälle.

Torstai 22.03.2018

Hiljaisempi loppuviikko kyseessä, joten torstaina ei ollut mitään suurempia tavoitteita. Tarkoituksena lähinnä edelleen perehtyä odoon rakenteeseen.

Päivä meni lueskellessa odoon dokumentaatiota, etenkin odooseen kuuluvan template engine QWebin tutkimiseen. Suurimpana vaikeutena odoon kanssa minulla tuntuisi olevan löytää oikeiden template kutsujen löytäminen. Esimerkit dokumentaatiossa usein tuntuvat kovin suuntaa antavilta, eivätkä ainakaan sellaisenaan päde. Tämä taas aiheuttaa ongelmia millä saan ”hajotettua” yksittäisiä sivuja odoosta, kun käytän odoosta löytyvää online editoria tiettyjen sivujen muokkaamiseen.

Päivälle ei ollut kummoisiakaan tavoitteita, mutta odoon dokumentaation ja rakenteen tutkiminen osaa kyllä omalla tavallaan tuskastuttaa.

Perjantai 23.03.2018

Päivän tavoitteena oli korjata aikaisemmin tekemäni filtteri Optiwise sovelluksessa, sekä lisätä muutama graafikon toimesta päivitetty logo.

Kyseessä oli filtteri, missä voit checkboxien avulla valita useamman valinnan, minkä perusteella backendistä haetaan tietoa. Checkboxien rastittaminen ja välitön tiedon haku toimi kyllä oikein, mutta kun suljit valinta laatikon, ei sovellus enää muistanut mitkä checkboxit olivat valittuina. Tässä olin vain omaa huolimattomuuttani unohtanut asettaa tarkistuksen onko filtterin valintoja tarkistettu vai ei. Seuraavaksi ongelmaksi muodostui se, että kun kyseistä filtteriä käytti, ei sivulla ollut paginointi toiminnallisuus toiminut sen kanssa. Tätä varten jouduin päivittämään serviceä missä tämän filtterin tulokset haetaan ja lisäämään sinne limit ja offset parametrit. Sen lisäksi jouduin lisäämään itse komponenttiin tarkistuksen, millä katsottiin, käytetäänkö juuri tätä filtteriä, vai jotain muita sivulla olevista filttereistä, sillä muiden filtterien vastaukset tulevat eri rajapinnasta. Nämä muutokset tehtyäni lisäsin ne gittiin ja tuotantopalvelimellemme. Testasin toiminnallisuutta vielä tuotantopalvelimella ja se tuntui toimivan niin kuin pitääkin.

Seuraavaksi päivitin graafikon antamat uudet logot sivulle, sekä lisäsin erillisen linkin ”yhteenveto” linkin dashboard näkymään, koska asiakkaalta oli sellainen toive tullut.

Onnistuin päivän tavoitteissani hyvin, mutta kaikki tehdyt asiat olivat jo ennalta opittuja.

Viikkoanalyysi

Tekemisen suhteen melko hiljainen viikko. Oli kyllä hyvä tutustua Odoon toimintaan vähän syvällisemmin. En myöskään aiemmin ole kirjoittanut docker-compose tiedostoja, vaikka dockeria käytänkin oikeastaan päivittäin työssäni. Oli kiva huomata, että Odoon

testiympäristön pystyttäminen dockerin päälle oli niinkin helppoa ja docker ohjeissa ei ollut mitään valittamista.

Docker itsessään on kyllä erinomainen työkalu, sillä pystyy erittäin nopeasti pystyttämään toimivia kehitys ja tuotantoympäristöjä, mikä tekee ainakin kehittämisestä erittäin miellyttävää. On myös erittäin käytännöllistä, että dockerin avulla jokaisella kehittäjällä on samanlainen kehitysympäristö, riippumatta siitä minkälainen ympäristö heillä on käytössään. Tämän avulla pystyy välttämään monia ympäristöihin liittyviä ongelmia. Dockerin voima on siinä, kuinka sen luonnollinen työnkulku mahdollistaa sovelluksien kulkemisen läpi niiden koko elinkaaren, suunnittelusta eläkkeelle siirtymiseen, yhdessä ekosysteemissä (Matthias & Kane 2015, 188).

OBOE ja filtteri bugin selvittäminen olivat myös omalla tavallaan mielenkiintoisia ja vähän erilaisia haasteita, kuin kohtaamani bugit yleensä.

3.7 Seurantaviikko 7

Maanantai 26.03.2018

Päivän tavoitteena oli poistaa muutama valinta ml filtteristä, koska backendin toiminnallisuutta muutettiin hieman.

Backendistä poistettiin muutamia tuloksia, joten kyseisten tulosten näyttäminen piti poistaa myös front-endin ml filtteristä. Itse poistaminenhan oli erittäin nopea operaatio, mutta sitten ilmenikin ongelma liittyen checkboxien raksimiseen, eli filtterien valintaan. Tietyissä tilanteissa kun valitsi yhden filtterin, saattoi kaksi eri filtteriä tulla samalla painalluksella, mikä ei tietenkään ollut haluttu toimintamalli. Syyksi tähän ilmeni käyttämäni for looppi, joka indeksin perusteella päivitteli valittuja filttereitä. Muutin tämän käyttämään Javascriptin find funktiota for looppauksen sijaan ja sain ongelman sillä korjattua.

Ei ollut mikään tapahtumarikas päivä, mutta saavutin tavoitteet ja onnistuin vielä korjaamaan löytyneen buginkin. Oli myös kiva käyttää harvemmin käyttämääni find funktiota.

Tiistai 27.03.2018

Päivän tavoitteena oli saada itselleni toimiva ZeeCore testiympäristö, missä voisin alkaa kehittää ZeeCore laitteen web käyttöliittymää. Sekä tietysti tutustua ZeeCoren nykyiseen koodipohjaan.

ZeeCoren kehittäminen lokaalisti on vähän ongelmallista, koska sitä ei saa ainakaan kovin helposti pyörimään dockerin päälle. Tästä johtuen sen kehittäminen tapahtuu suoraan laitteessa itsessään. Minulle allokoitiin omaan käyttöön yksi toimiston testi ZeeCoreista, missä voisin aloittaa kehittämisen. ZeeCoreen ja sen tiedostoihin pääsee käsiksi SSH yhteydellä. Totesin kuitenkin nopeasti, että SSH:n / sftp:n yli tiedostojen siirtely on kovin tuskaista ja lähdin selvittämään edes vähän kätevämpää ratkaisua tämän toteuttamiseen. Googlettelin tovin ratkaisua ongelmaani ja löysin tavan mountata etä kansion paikalliseen koneeseeni. Tähän löysin ohjeet digitaloceanista. Ensin piti asentaa SSHFS, minkä pystyi Ubuntulla asentamaan ihan apt-getillä. Tämän jälkeen piti luoda kansio mihin mountataan ZeeCoren tiedostojärjestelmä. Sitten ajamaan komennon *sudo sshfs -o allow_other root@xxx.xxx.xxx.xxx:/ /mnt/ZeeDev.* Tämän jälkeen pystyikin avaamaan editorilla mountatun tiedostojärjestelmän ja alkaa tutustumaan ZeeCoren koodipohjaan.

En ollut aikaisemmin ollut tekemisissä ZeeCoren kanssa, joten kaikki tähän päivään liittyvä oli uutta minulle. En myöskään ollut aikaisemmin käyttänyt SSHFS:ää, joten opin myöskin uuden kätevän työkalun käytön. Tutustuin myös vähän ZeeCoren koodipohjan rakenteeseen, jotta voisin alkaa huomenna työskentelemään sen parissa.

Keskiviikko 28.03.2018

Päivän tavoitteena on tutustua vielä paremmin ZeeCoren koodipohjaan, sekä aloittaa muutoksien teko ZeeCoren web käyttöliittymän ulkoasuun.

Ensimmäisinä muutoksen kohteina on menu ja eräänlaiset widgetit, mitä ZeeCoren web käyttöliittymässä on. Graafikkomme teki muutamia layout ehdotelmia, minkä pohjalta alan muutoksia tekemään. Ensimmäisenä ongelmana oli saada selville, miten kyseisten toiminnallisuuksien luominen nykyisessä koodissa toimii. ZeeCoren käyttöliittymä on toteutettu PHP:lla ja jonkinlaisella template enginellä, mistä en sen tarkempaa tieto kollegoilta saanut. En ole koskenut PHP koodiin muutamaan vuoteen, joten otti aikansa ymmärtää miten esim. menu luotiin. Pääsin kuitenkin pikkuhiljaa sisälle siitä, miten koodi toimii ja lähdin aloittamaan menun ulkoasun muokkaamista. Alkuperäin menu oli toteutettu vain vaihtelemalla eri kuvien välillä, riippuen mikä oli valittuna. Poistin siis koko olemassa olevan menun ja aloin kirjoittamaan sitä uusiksi. Menun muotoilut tulisin taas toteuttamaan CSS gridillä. Kerkesin päivän loppuun mennessä saamaan alustavan menun rakenteen toteutettua, mutta huomiselle jäisi vielä tarkempi fonttien ja menun kuvien asettelu.

Onnistuin päivän tavoitteissa omasta mielestäni hyvin. Pääsin paremmin sisälle siitä, miten ZeeCoren web käyttöliittymä on toteutettu ja pääsin aloittamaan siihen liittyvät muokkaukset.

Torstai 29.03.2018

Päivän tavoitteena saada ZeeCoren uusi menu valmiiksi, sekä muokata widgettien ulkoasu vastaamaan enemmän graafikon näkemystä.

Jatkoin siis siitä mihin eilen jäin, eli menusta. Lisäsin menun linkkeihin myös graafikon antaman logot, sekä graafikon määrittelemän fontin. Tämän jälkeen lähdin selvittämään miten käyttöliittymässä olevat widgetit oikein rakentuvat. Widgeteille löytyi oma template tiedosto, mutta siinä ei näkynyt kaikki CSS luokat, mitä esim. Chromen inspectorilla kyseisestä templatestä löytyi. Kysyin kollegalta vähän tietoa mistä tämä johtuisi ja hän epäili sen johtuvan vanhahkosta jQuery ui:n versiosta mikä ZeeCoressa on käytössä. En keksinyt mitään suoraa ratkaisua ylimääräisten CSS luokkien poistamiseen, joten päätin vain korvata kyseisten luokkien CSS määritykset erillisessä CSS tiedostossa. Näin sain muokattua widgettien ulkoasua vastaamaan graafikon näkemystä paremmin.

Onnistuin päivän tavoitteissa omasta mielestäni hyvin. Sain tehtyä ZeeCoren menun valmiiksi, sekä muokattua widgetit näyttämään ainakin oikeammalta. En ikävikseni saanut selville mistä / miten ylimääräiset CSS luokat ilmestyvät, mutta onneksi kuitenkin pystyin korvaamaan näiden luokkien määritykset erillisessä CSS tiedostossa.

Perjantai 30.03.2018

Vapaapäivä, pitkäperjantai.

Viikkoanalyysi

Vähän erilainen viikko kyseessä, pääsin tutustumaan yrityksemme IOT laitteen eli ZeeCoren web käyttöliittymän sielunelämään. Suurimpia ongelmia oli PHP:n käyttö pitkän tauon jälkeen, sekä selvittää miten pystyy kätevästi editoimaan koodipohjaa suoraan palvelimella ilman, että tiedostoja tarvitsee erikseen siirtää erillisellä scp työkalulla.

Yksi viikon haasteista oli päästä sisälle jQuery ui:n toimintaan. Käytössä oli vielä erittäin vanha versio jQuery ui:sta, joten sen toiminta vähän hämmensi itseäni, varsinkin, kun olen tottunut työskentelemään enemmän angularin parissa. Sain kuitenkin lopulta muokattua

tyylejä enemmän haluamani kaltaiseksi, joten olen tyytyväinen aikaansaannoksiini ZeeCoren käyttöliittymän parissa.

Oman haasteensa toi myös PHP:n maailma, en ollut todellakaan käyttänyt PHP:ta pitkään aikaan. En myöskään hirveästi nauti PHP:llä ohjelmoinnista, koska paikka paikoin se on hyvin vaikealukuista. Tämä johtuu pääasiassa siitä, että PHP koodin sisään on saatettu upottaa satunnaisia pätkiä html ja Javascript koodia. Toki nykyaikaisemmassa PHP kehityksessä asiat saattavat olla toisin, mutta ZeeCoren ollessa kyseessä on koodipohja jo erittäin vanha.

Vanhoihin monoliitti järjestelmiin tutustuminen on kuitenkin ollut hyvästä, sillä niiden kautta pystyn paljon paremmin ymmärtämään mikropalveluiden tuomat hyödyt. Selvimpiä hyötyjä esimerkiksi Optiwise sovelluksen eduksi verrattuna ZeeCoreen on se, että Optiwisessa front-end ja backend ovat kaksi täysin erillistä sovellusta. Ne toki keskustelevat keskenaan REST-rajapintojen kautta, mutta kumpikaan ei sinänsä vaadi toista toimiakseen. Tämä taas edesauttaa sitä, että molemmat on voitu toteuttaa juuri tarkoitukseen sopivilla kielillä. Artikkelissa "Mikropalvelut nousivat hypen huipulle - mitä hyötyä niistä on?" (Pertti Hämäläinen 2016.) avataan omasta mielestäni mikropalvelujen hyötyjä erittäin hyvin. Esimerkiksi juuri ohjelmoijan näkökulmasta ne selkeyttävät tehtäviä töitä, koska on paljon helpompaa hallita ja kehittää useampaa pientä helpommin ymmärrettävää kokonaisuutta, kuin yrittää kehittää yhtä massiivista monoliittia, joka pitää kaiken sisällään kaiken mahdollisen.

3.8 Seurantaviikko 8

Maanantai 16.04.2018

Päivän tavoitteena oli päivittää ZeeCloudin beta instanssissa olevan esimerkki kuvan istuvuutta ruudulle.

Asiakkaalle näkyvässä beta instanssissa oli graafikon tekemä placeholder kuva ominaisuuksista, mitä voisimme ottaa käyttöön asiakkaan ympäristössä. Ongelmana oli, kuvan loppuosan pois leikkaantuminen. Tähän ongelmaan oli muutama eri syy. Pääasiallisena syynä se, että ZeeCloudin widgetit, mistä myös kyseinen kuva tuli, ovat jokainen omia iframejaan. Olin aiemmin päivittänyt iframejen CSS:ää, mikä nyt näytti aiheuttavan kyseisen ongelman. Toisena syynä oli se, että kuva oli yksinkertaisesti määritetty liian suureksi Djangon admin käyttöliittymässä. Poistin siis CSS määrityksen mikä aiheutti widgettien leikkaantumisen, jos ne olivat tarpeeksi leveitä, sekä kävin muuttamassa Djangon admin

käyttöliittymästä kuvan leveyttä vähän pienemmäksi, jotta sitä ei tarvitsisi turhaan scrollata sivuttaissuunnassa.

Onnistuin päivän tavoitteissa hyvin, sillä sain kaiken toimimaan niin kuin pitääkin. Suurin ongelma oli löytää kuvan leikkaamisen aiheuttava CSS määritys, mikä olikin vähän hämmentävä, koska siinä oli käytetty CSS:n view width ja calc toiminnallisuuksia.

Tiistai 17.04.2018

Viikosta näyttäisi tulevan vähän hiljaisempi, joten päivän tavoitteena oli jatkaa jo aikaisemmilla viikoilla aloittamiani ZeeCoren web käyttöliittymän ulkoasun päivityksiä.

Olin jo aiemmin päivittänyt ZeeCoren navigaation näyttämään modernimmalta ja nyt vuorossa oli etusivun widgettien päivittäminen. Aloitin siis vertaamalla graafikolta saamiani kuvia ZeeCoren nykyiseen näkymään. Tämän jälkeen kävin yksinkertaisesti Chromen inspectorilla läpi eri html elementtejä ja vaihtelin niiden värejä ja kokoja vastaamaan tavoitetilaa. Tällä tavalla edeten aloin pikkuhiljaa kirjoittaa uusia CSS luokkia missä asettelin kyseisille elementeille uudet määritykset, jotka vastasivat inspectorissa tekemiäni muutoksia.

Edistyin ZeeCoren etusivun widgettien ulkoasun päivittämisessä, joten pikkuhiljaa tämäkin ulkoasun päivitysprojekti etenee kohti tavoitetilaa. Tämän päivän tehtävissä ei ollut merkittäviä haasteita. Pienenä haasteena sanoisin olevan sen, että käytössä ei ole mitään valmista ulkoasu kirjastoa kuten esim. Bootstrap vaan tietyissä tilanteissa joudun vähän, kuin keksimään pyörää uudelleen.

Keskiviikko 18.04.2018

Graafikkomme oli saanut tehtäväksi käydä vielä läpi ZeeCloudiin viime aikoina tehtyjä muutoksia ja hänen oli tarkoitus katsoa, ilmeneekö niissä mitään merkittäviä virheitä mitä minun pitäisi korjata. Tästä sainkin tehtäväksi pieneltä kuulostavan asian. ZeeCloudin navigaatio palkin ikonit ”pomppivat”, kun vaihdettiin pienennetyn ja ison navigaatiopalkin välillä. Myöskin pienennetyssä navigaatiossa olevan logon sijaintia täytyisi vähän muuttaa.

Piti itse oikein hieraista silmiä ja katsoa ilmeneekö tätä ”pomppimista” myös minulla ja kyllähän sitä ilmeni. Kyseessä oli silmämääräisesti katsottuna vain muutaman pikselin heitto, mikä kyllä alkoi ärsyttämään, kun sen olemassaolon tiedosti. Tarkemmin tutkittuna heitto oli noin 0.5 pikseliä. Tein kaksi muutosta korjatakseni asian. Muutin pienennetyn

sivupalkin logoa vähän isommaksi ja lisäsin paddingia sen yläpuolelle. Samalla tuli tehtyä graafikon haluama logon sijainnin pieni muutos.

Päivän tavoitteissa tuli onnistuttua hyvin, mutta ikonien pomppimisen selvittäminen ja korjaaminen tuotti yllättävän paljon vaivaa. Päivässä ei sinänsä ollut mitään uutta mitä olisin oppinut, mutta en ehkä aiemmin ole joutunut kuluttumaan niin paljoa aikaa noinkin pienessä ongelmassa.

Torstai 19.04.2018

Päivän tavoitteena oli saattaa ZeeCloudin uusi ulkoasu sellaiseen kuntoon, että sen voisi luovuttaa taas asiakkaan tarkastettavaksi. Aiemmin olin muokannut jo kirjaudu sisään sivun täysin uudennäköiseksi, mutta rekisteröidy sivun muokkaaminen oli täysin unohtunut.

Aloitin siis vertaamalla kirjautumis- ja rekisteröitymis- sivujen ulkonäköä ja eroavaisuuksia. Periaatteessa molemmat sivut pystyisivät käyttämään lähes samaa asettelua. Ainoa eroavaisuus oli, että rekisteröitymis- sivussa on enemmän täytettäviä kenttiä. Toisena erona oli, että rekisteröitymis- sivun kentät tulevat Djangon viewistä, eikä templatesta, kuten kirjautumis- sivulla. Muokkasin siis ulkoasun suurimmaksi osaksi vastaamaan kirjautumissivua, mutta aivan täydellisesti en pystynyt sitä mallintamaan, koska en pystynyt itse templatessa muokata viewistä tulevia input kenttiä.

Onnistuin muokkaamaan rekisteröitymis- sivun pääpirteittään sellaiseksi, kuin oli tarkoituskin. Teknisistä rajoitteista johtuen en saanut sitä aivan identtiseksi kirjautumis- sivun kanssa, mutta tarpeeksi hyväksi kuitenkin. Tuli myös vähän tutustuttua Djangon vieweihin mihin en aiemmin ollut juurikaan perehtynyt.

Perjantai 20.04.2018

Perjantain tavoitteena oli päivittää ZeeCoressa ja ZeeCloudissa näkyviä on/off ikoneja uudempiin versioihin, sekä laittaa ZeeCoreen tekemäni muutokset Git repositoryyn.

Alotin siis päivittämällä uudet ikonit ZeeCloudiin, mikä vaikutti aluksi ongelmalliselta, sillä en löytänyt miten kyseisiä kuvatiedostoja kutsutaan. Tovin etsimisen jälkeen löysin Djangon modelin, missä määriteltiin mitä ikoneja, milloinkin kutsutaan, joten muokkasin kyseisen modelin hakemaan uusia ikoneja ja homma oli sillä selvä. Tämän jälkeen päivitin ZeeCoresta löytyvät vastaavat ikonit, mikä onnistuikin huomattavasti helpommin.

Tähän asti olin tehnyt kaikki ZeeCoreen liittyvät muutokset, suoraan itse ZeeCore laitteella, tallentamatta muutoksia mihinkään Git repositoryyn. Otin siis uusimman ZeeCoresta löytyvän Git haaran, minkä pohjalta loin uuden Git haaran ja kopion tekemäni muutokset kyseiseen haaraan ja puskin sen Gitlabiin.

Pääsin tutustumaan ZeeCloudin Djangon modeleihin, sekä vihdoinkin laitettua ZeeCoreen tekemäni muutokset talteen, mikä oli erittäin hyvä asia, jotta tekemäni muutokset eivät vahingossa pääse katoamaan vaikkapa yllättävästä laiterikosta johtuen.

Viikkoanalyysi

Vaikka kyseessä olikin vähän hiljaisemman oloinen viikko, sain silti omasta mielestäni hyvän määrän muutoksia tehtyä. Sain myöskin vähän enemmän syvennettyä ymmärrystäni Djangosta, liittyen modeleihin ja vieweihin. Sain myöskin vihdoinkin laitettua ZeeCoreen tekemiäni muutoksia Git repositoryyn, mikä onkin erittäin tärkeää. Ei pelkästään sen takia, että tekemäni muutokset pysyisivät tallessa, vaan myöskin sen takia, että commit historiaa alkaisi kertyä tekemilleni muutoksille. Olen muutenkin huomannut versionhallinnan olevan tärkeää, jotta esimerkiksi mahdollisien bugien ilmentyessä pystytään nopeasti palaamaan aiemmin käytössä olleeseen versioon sovelluksesta, sekä mahdollisesti paikantamaan helpommin ongelman aiheuttanut kohta. Tässä auttaa varsinkin se, että commit viesteissä on jotain järkeä.

Tärkeää on kirjoittaa merkityksellinen aihe commit viestille. Sen tarkoitus on selvittää mitä kyseinen commit sisältää. Tässä kannattaa välttää teknisiä yksityiskohtia, jotka muut kehittäjät kyllä ymmärtävät avattuaan kyseisen koodinpätkän. Keskity isoon kuvaan ja muista, että jokainen commit viesti on lause versionhallinnassa. Ajattele kuten muutoslokin lukija ja yritä kirjoittaa kaikkein ymmärrettävin viesti hänelle, älä itsellesi. Käytä nykymuotoa ja kirjoita lause, mikä pitää sisällään maksimissaan viisikymmentä merkkiä. (Santacroce, Olsson, Voss & Narębski 2016, 104.)

Toinen asia, mikä on tärkeä muistaa, on aloittaa commit viestit isoilla kirjaimilla. Älä lopeta lauseita pisteellä, ne ovat hyödyttömiä ja jopa vaarallisia. (Santacroce ym. 2016, 104.)

On myös toinen asia mihin minun olisi hyvä kiinnittää enemmän huomiota ja se on committien laajuus. Useasti tehdessäni muutoksia varsinkin vanhempaan koodipohjaan, tulee korjattua monia pieniä asioita mitkä eivät välttämättä liity toisiinsa ja tämän jälkeen puskea versionhallintaan kaikki muutokset saman commitin alle. Tämä aiheuttaa sellaisen

ongelman, että jos kyseissä commitissa onkin jonkinlainen virhe, mikä pitäisi korjata, voi olla hyvinkin vaikeaa löytää mistä se oikein aiheutuu, koska commit on niin laaja.

Ainoa tapa ratkaista kyseinen ongelma on tehdä vain yksi muutos committia kohden. Se vaikuttaa helpolta, mutta sitä varten ei ole mitään työkaluja. Kukaan muu kuin sinä itse, ei voi vaikuttaa siihen. Se vaatii itsekuria, mikä on kaikkein puutteellisin hyve luovien ihmisten, kuten ohjelmoijien keskuudessa. (Santacroce ym. 2016, 99.)

3.9 Seurantaviikko 9

Maanantai 23.04.2018

Päivän tavoitteena oli jatkaa ZeeCoren ulkoasuun liittyvien päivityksien parissa.

Kerkesin vasta aloittamaan ZeeCoren parissa työskentelyn, kun minulle tultiin sanomaan, että ZeeCloudissa oleva kuva ”leikkaantuu” vieläkin pienillä resoluutioilla. Olin omasta mielestäni korjannut kyseisen ongelman jo viime viikolla, mutta se oli palannut syystä tai toisesta. Pikaisesti tutustuttuani ongelmaan tajusin, että olin vain unohtanut committaa kyseisen muutoksen. Joten committasin ja pushasin muutoksen develop haaraan ja palasin taas ZeeCoren pariin. Muutin siis muutaman eri formin ulkonäköä vastaamaan graafikon enemmän graafikolta saamiani kuvia liittyen uuteen ulkoasuun. Teknisistä rajoitteista johtuen en saanut formeja vastaamaan täydellisesti graafikon näkemystä. Sain kuitenkin päivitettyä niitä merkittävästi paremman näköiseksi, mistä on hyvä jatkaa.

ZeeCoren web käyttöliittymässä on paljon pieniä ulkoasuun liittyviä bugeja, joita on välillä vaikea löytää. Suurin osa näistä johtuu selkeästi vain vanhoista käytössä olevista kirjastoista, kuten jQuery ui:sta. Tällä hetkellä ei ikävä kyllä ole mahdollista päivittää käytössä olevia kirjastoja, koska se saattaisi hajottaa olemassa olevia toiminnallisuuksia. Sain kuitenkin tehtyä ulkoasu päivityksiä ZeeCoreen ja näin vietyä ulkoasun modernisointia eteenpäin, joten päivä oli kaiken kaikkiaan onnistunut.

Tiistai 24.04.2018

Päivän tavoitteena modernisoida muutamien taulukkojen ulkoasua ZeeCoren web käyttöliittymässä. Taulukoiden olisi tarkoitus näyttää samanlaiselta kuin bootstrapistä löytyvä table-striped määrittely.

Aloitin siis taulukoiden kanssa työskentelyn ja koska ZeeCoressa ei ole bootstrappia käytössä aloin jo keksimään pyörää uudelleen ja tekemään vastaavanlaista muotoilua taulukoille, kuin mitä bootstrapin table-striped määrittely tekee. Onnekseni en ehtinyt vielä hirveästi tehdä, kun tajusin kysyä kollegaltani, voisinko lisätä bootstrapin ZeeCoreen. Kollegani näytti vihreää valoa, kunhan vain käyttäisin minifioitua versiota bootstrapista. Myöskään cdn:n käyttö ei ole mahdollista ZeeCoren ollessa kyseessä. Latasin siis minifioidun bootstrapin ja siihen tarvittavan minifioidun version jQuerystä. Siirsin nämä ZeeCorelle ja huomasin heti, että jotain hajosi. Etusivun käyttöliittymä meni aivan omituisen näköiseksi. Ajattelin heti, että tämä johtuu lisäämästäni jQuerystä, joka varmaankin sisältää päällekäisyyksiä jQuery ui:n kanssa. Poistin jQueryn ja kaikki näytti taas siltä, kuin ennekin. Onneksi bootstrap ei tarvitse jQueryä, kuin osassa toiminnallisuuksistaan, joten päätin jättää jQueryn pois ja kokeilla toimisiko taulukkojen table-striped määritys ilman jQuerya. Taulukot näyttivät kyseisen määrityksen jälkeen juuri oikeilta, joten olin tyytyväinen ratkaisuuni.

Onnistuin pienien haasteitten jälkeen päivittämään ZeeCoressa olevat taulukot näyttämään siltä, kuin graafikkomme oli ajatellut. Joka päivä työskennellessäni vanhan koodipohjan parissa huomaan, kuinka omalla tavallaan vaikeaa voi pienienkin muutoksien tekeminen olla. Mutta ei auta kuin purra hammasta ja yrittää päästä haluttuihin tuloksiin hajottamatta mitään toiminnallisuuksia.

Keskiviikko 25.04.2018

Päivän tavoitteena jatkaa ZeeCoren ulkoasuun liittyviä päivityksiä. Spesifimmin sanottuna web käyttöliittymän taustaväri hajoaa mielenkiintoisesti osalla sivuista ja olisi tarkoitus selvittää mistä tämä aiheutuu ja korjata se.

Otin auki yhden sivun missä taustaväri hajoaa ja aloin chromen developer työkaluilla tutkimaan html elementtejä löytääkseni ongelman aiheuttavan kohdan. Pitkän etsimisen jälkeen totesin ongelman jollain tavalla johtuvan body elementistä, kun vaihdoin Chromen inspectorilla bodyn taustaväriä, sain taustan näyttämään oikeanlaiselta. Tämän jälkeen tein saman muutoksen CSS tiedostoon ja yllätyksekseni se ei korjannut ongelmaa. CSS tiedostoon tekemäni muutos ei jostain syystä päivittynyt ja kyseessä ei ollut mikään välimuistista johtuva ongelma. Lopulta tulin siihen tulokseen, että ongelma tulee Javascriptin, todennäköisesti jälleen kerran jQuery ui:n puolelta. Päätin siis kokeilla muuttaa body elementin taustaväriä jQueryn $(document).ready() funktion sisällä. Tämä muutos todellakin toimi ja sain sillä ratkaistua ongelmani ja sivut näyttivät siltä kuin pitääkin.

Onnistuin päivän tavoitteissa täydellisesti. Kohtasin ongelman mikä oli omasta mielestäni erittäin hämmentävä ja vaikea selvittää, mutta pienellä kekseliäisyydellä selvisin siitä.

Torstai 26.04.2018

ZeeCoren ulkoasun muutoksien tekeminen jatkuu. Tavoitteena yrittää korjata erinäisten nappien ulkonäköä, koska ne eivät enää sovi päivitettyyn taustaan.
Ensimmäiseksi korjasin muutamien nappien värityksiä vastaamaan uutta graafista ilmettä. Tämän jälkeen huomasin, että osaan napeista tekemäni muutokset eivät vaikuttaneet, joten aloin Chromen inspectorilla tutkia mistä tämä voisi johtua. Huomasin, että kyseisillä napeilla oli joitain omia CSS luokkia, mitkä jälleen kerran oletettavasti tulivat jQuery ui:n puolelta. Ratkaisuna tähän ei auttanut kuin yli kirjoittaa kyseiset muutokset ja tällä tavalla korjata nappien ulkoasu. Viimeiseksi korjattavaksi jäi nappi mikä oli kokonaan kadonnut erään formin taustaan, koska taustaväri oli muutettu. Ongelmaksi tässä muodostui se, että kyseinen nappi olikin jQuery ui:sta tuleva kuva ja tätä ei pystynytkään oikein järkevästi muuttamaan. Päädyin siis tekemään väliaikais- ratkaisun, millä sain napin erottumaan taustasta, mutta se ei ollut mikään kovin kaunis ratkaisu.

Onnistuin päivän tavoitteissani pääosin hyvin, ainoaksi murheeksi jäi yksittäinen nappi mitä en saanut muokattua sen näköiseksi, kuin halusin. Tämän pariin tulen vielä palaamaan myöhemmin.

Perjantai 27.04.2018

Päivän tavoitteina edelleen jatkaa ZeeCoren ulkoasun parissa, sekä käydä pienimuotoinen viikkopalaveri missä käydään läpi mitä kenelläkin on työn alla.

Päivä alkoi palaverilla käymällä vähän läpi mitä kenelläkin on työn alla ja miten asiat edistyvät. Samalla kollegani kertoi, että tulisimme mahdollisesti ensiviikolla dokumentoimaan optiwise sovelluksen front-endin. Tarkoituksena kirjoittaa muutaman a4:sen verran dokumentaatiota, missä käydään läpi sen toiminnallisuuksia, rakennetta ja miten sen ympäristö pystytetään. Palaverin jälkeen jatkoin ZeeCoren parissa, lähinnä korjailemalla pieniä kauneusvirheitä.

Päivälle ei ollut mitään isompia tavoitteita, mutta oli kiva kuulla, että tulemme vähän dokumentoimaan optiwisen front-endiä, koska tällä hetkellä sen dokumentaatio on melko olematonta. Sain myöskin taas edistettyä ZeeCoren ulkoasua, joskin pienin askelin.

Viikkoanalyysi

Viikko keskittyi lähes kokonaan ZeeCoren ulkoasun muokkaamiseen. Viikkoon sisältyi silti suhteellisen paljon ongelmia, mitä pääasiassa johtuivat vanhoista käytössä olevista kirjastoista ja niiden aiheuttamista outouksista. ZeeCoren web käyttöliittymän ollessa kyseessä on ongelma kohtien paikantaminen myös joskus erittäin vaikeaa. Tämä johtuu toki vanhoista kirjastoista, mutta myös siitä, että käytetty kieli on php. Php:tä en ole edelleenkään käyttänyt kovin paljoa, eikä koodin ikä tee siihen puuttumisesta yhtään helpompaa.

Astuminen tuntemattoman koodin sisään, varsinkin legacy koodin, voi olla pelottavaa. Ajan kuluessa jotkut ihmiset tulevat immuuniksi pelolle. He kehittävän itseluottamusta kohdatessaan ja tappaessaan monstereita koodissa uudestaan ja uudestaan, mutta on erittäin rankkaa olla pelkäämättä. Jokainen kohtaa joskus demoneja, joita he eivät voi tappaa. Jos jäät murehtimaan sitä ennen kuin edes katsot koodia. Tekee se siitä vaikeampaa. Et koskaan tiedä onko tekemäsi muutos helppo vai viikon kestävä hiusten kiskomis harjoitus, joka jättää sinut kiroamaan järjestelmää. (Feathers 2005, 338.)

3.10 Seurantaviikko 10

Maanantai 30.04.2018

Päivälle ei mitään suurempia tavoitteita, tarkoituksena lähinnä käydä läpi pieniä jäljellä olevia korjauksia ulkoasuun liittyen.

Vappuaatosta johtuen vähän hiljaisempi päivä toimistolla, enkä itsekkään ollut aivan täyttä päivää paikalla. Löysin muutamia pieniä korjattavia asioita, kun osassa widgeteissä näkyvistä teksteistä oli hävinnyt taustavärin vaihdosta johtuen. Nämä nyt oli kuitenkin hyvinkin helppo korjata.

Lyhyempi päivä ilman kummoisempia tavoitteita, onnistuin kuitenkin muutamia pikku vikoja korjaamaan, joten kaiken kaikkiaan ihan hyvä päivä.

Tiistai 01.05.2018

Vapaapäivä, vappu

Keskiviikko 02.05.2018

Päivän agendassa on käydä palaveri erp järjestelmä Odoo:n online proposals moduliin varsinaisesta käyttöönotosta, sekä jatkaa ZeeCoren käyttöliittymän parissa työskentelyä.

Aloitin päivän siis jälleen kerran käymällä läpi löytyisikö ZeeCoresta enää mitään isompia, tai edes vähän pienempiä bugeja. ZeeCoren web käyttöliittymän ulkoasu alkaa kyllä näyttämään melko valmiilta, enkä oikeastaan löydä enää mitään korjattavia bugeja siihen liittyen.

Pidimme loppupäivästä esimieheni kanssa pienen palaverin liittyen
odoon online proposals moduuliin. Itse moduulinhan otin tuotantopalvelimella käyttöön jo muutama viikko sitten, mutta sitä ei varsinaisesti ole vielä käytetty mihinkään. Esimieheni halusi tietää miten kyseisellä moduulilla tehtäviä tarjous templateja pystyisi tekemään ja muokkaamaan. Kävimme hänen kanssaan läpi miten uusia tarjouksia pystyisi luomaan ja miten niiden sisältöä pystyy muokkaamaan. Yksi ongelma ilmeni templaten ulkoasua muokatessa, esimieheni ei syystä tai toisesta pystynyt editoimaan sitä. Tarjous pohjat näyttivät myös väärän yrityksen logoa, joten tämäkin ongelma pitäisi selvittää.

ZeeCoren web käyttöliittymän ulkoasu alkaa näyttämään erittäin valmiilta ja sen suhteen ei tarvinnutkaan tänään oikeastaan mitään muutoksia enää tehdä. Loppupäivän palaverissa tuli ilmi muutamia seikkoja mitä joutuisin huomenna käymään läpi, mikä olikin erittäin hyvä asia. On myös kiva saada online proposals moduulin käyttöönottoa taas eteenpäin.

Torstai 03.05.2018

Tämän päivän tavoitteena on käydä läpi Odoo:ssa eilen ilmenneitä ongelmia ja yrittää selvittää sekä korjata ne. Tarkemmin sanottuna tarkoituksena oli siis selvittää miksi esimieheni ei pystynyt muokkaamaan tarjous templateja ja miksi tarjouksessa näkyi väärän yrityksen logo.

Ensimmäiseksi aloin siis omassa lokaalissa testiympäristössäni testailemaan, että pystynkö muokkaamaan tarjous templatejen ulkoasuja. Pystyin muokkaamaan ulkoasuja ongelmitta, joten tuntui vähän oudolta, että esimieheni ei niin pystynyt tekemään. Testimielessä loin siis uuden käyttäjän lokaaliin testiympäristööni, testatakseni onnistuisiko sillä templatejen ulkoasun muokkaaminen. Myös uudella käyttäjällä ulkoasun muokkaaminen onnistui, joten olin jo vähän enemmän ihmeissäni. Tämän jälkeen päätin kirjautua admin tunnuksilla tuotantopalvelimelle ja tutkia olisiko esimieheni käyttöoikeuksissa jotain hämärää. Vika todellakin löytyi käyttöoikeuksista, sillä esimiehelläni ei ollut määriteltynä mitään käyttöoikeuksia templatejen muokkaamiseen liittyen. Päättelin tämän johtuvan siitä, että

hänen tunnuksensa oli luotu jo kauan ennen, kuin online proposals moduulia oli edes asennettu ja moduulin asentaminen ei tietenkään automaattisesti luonut olemassa oleville käyttäjille tarvittavia oikeuksia.

Väärän logon näkyminen korjaantui oikeastaan samalla kertaa, kun annoin oikeat oikeudet esimiehelle, pystyi hän vaihtamaan logonkin oikeaksi.

Onnistuin päivän tavoitteissa hyvin, sillä sain tehtyä kaiken mitä pitääkin. Opin samalla myös hieman, miten odoo:ssa määritellään käyttäjille oikeuksia ja mitä erilaisia oikeuksia sieltä oikein löytyykään.

Perjantai 04.05.2018

Päivän tavoitteena oli selvittää pystyisikö ZeeCoren vanhan jQuery ui:n päivittämään uudempaan versioon. Tavoitteena oli myöskin käydä läpi suunnitteilla olevan sovelluksen arkkitehtuuria.

Heti aamusta kävimme kollegani kanssa läpi erään suunnitteluvaiheessa olevan sovelluksen arkkitehtuuria ja graafikkomme näytti meille sitä varten tekemiään luonnoksia. Pohdimme myös tulisiko tämän sovelluksen front-end toteuttaa angularilla ja minkälainen rakenne olisi hyvä kokonaan uudelle projektille.

Kollegani kanssa huomasimme molemmat myös, että angular oli päivittynyt versioon 6. Vaikka kyseinen versio päivitys ei sinänsä näyttänyt tuovan mitään järisyttäviä muutoksia, niin emme voi ainakaan Optiwise sovellusta alkaa heti päivittämään uusimpaan versioon. Ongelmana isommissa angular päivityksissä on se, että kirjastot laahaavat perässä. Käytössämme on monia kolmannen osapuolen kirjastoja, mikä aiheuttaa aina angularin version päivitysten yhteydessä suurta päänvaivaa ja tätä varten pitäisi keksiä joku ratkaisu.

ZeeCoren:sta tällä hetkellä löytyvä jQuery ui:n versio on 1.8, kun uusin versio on 1.12.1. Tarkoituksena on siis yrittää päivittää versio uudempaan. Ongelmaksi muodostuu se, että olen tehnyt jo paljon muutoksia ZeeCoren web käyttöliittymään ja tekemäni muutokset eivät välttämättä toimi enää näin ison versio päivityksen jälkeen. Mahdollisesti myös muita ennalta arvaamattomia asioita rikkoutunee päivityksen yhteydessä. Tästä huolimatta päätin valita ZeeCoressa käytössä olevan jQuery ui teeman ja alkaa kustomoimaan sitä jQuery ui:n sivuilla vastaamaan uutta graafista ulkonäköä.

Päivän tavoitteissa tuli onnistuttua puolittain. Kävimme onnistuneesti läpi suunnitteilla olevan sovelluksemme rakennetta, mutta en kerennyt kokeilemaan vielä uutta versiota jQuery ui:sta testi ZeeCorellani.

Viikkoanalyysi

Kyseessä oli vähän vaihteleva viikko, työskentelyä ZeeCoren ja Odoon parissa, sekä erilaisia pohdintoja liittyen sovellusten ylläpidettävyyteen ja arkkitehtuuriin. Mielenkiintoisimmaksi pohdinnaksi jäi kollegani kanssa käymäni keskustelu koodin mätänemisestä. Esimerkiksi puoli vuotta sitten tehtyä ja sen jälkeen unohdettua angular projektia on hyvin vaikea jatkaa nyt, koska itse framework ja projektissa mahdollisesti käytössä olevat kirjastot ovat päivittyneet niin paljon, että koodi ei välttämättä ole enää validia. Projektin toki saanee toimimaan, mutta siihen joutuu näkemään suhteettoman paljon vaivaa. Tämä samainen ongelma vähän vaivaa myös Optiwise sovellusta. Vaikka sovellus onkin aktiivisessa kehityksessä, mutta angularin merkittävän versio numeron päivittyessä on kirjastojen päivittäminen välillä todella tuskaista.

Oli kiva päästä myös välillä työskentelemään Odoon parissa ja tulikin taas opittua muutamia asioita käyttöoikeuksien määrittelystä. Tuli myöskin ylipäätään vähän palauteltua mieleen, miten Odoon kanssa työskennellään.

Myöskin oli mielekästä keskustella siitä, miltä vielä suunnitteluvaiheessa oleva sovelluksemme tulisi näyttämään ja mitä kaikkea se pitää sisällään. Sekä pohdiskella mitä kirjastoja siinä tulisi käyttää.

Hyvän suunnittelu laadun mittari on yksinkertaisesti vaadittavan vaivan mittaaminen täyttääksesi asiakkaan haluamat tarpeet. Jos tämä vaiva on vähäinen ja pysyy vähäisenä läpi koko järjestelmän elinkaaren, on suunnittelu ollut hyvää. Jos tämä vaiva kasvaa jokaisen uuden päivityksen myötä on suunnittelu huonoa. (Martin 2017, 35.)

4 Pohdinta ja päätelmät

Tämän luvun tarkoituksena on käydä läpi, miten olen kehittynyt seurantaviikkojen aikana, sekä käydä läpi isoimpia huomioita ja muutoksia työskentelytavoissani. Pyrin myös kuvailemaan uusia ratkaisutapoja ongelmiin, sekä käymään läpi seurantaviikkojen aikana oppimiani asioita.

4.1 Huomiot ja haasteet

Oma työnkuvani on laajentunut merkittävästi seurantaviikkojen aikana. Aloittaessani opinnäytetyön tekemisen oletin työskenteleväni suurimmaksi osaksi Optiwise sovelluksen parissa, kuten ennen opinnäytetyötänikin. Päädyin kuitenkin työskentelemään useamman eri sovelluksen parissa Optiwise sovelluksen lisäksi kehitin ZeeCloudia, ZeeCorea ja Odoota. Samalla jouduin poistumaan mukavuusalueeltani Angularin parissa ja tutustumaan myös PHP:lla ja jQueryllä työskentelyyn. Syvensin myös osaamistani Dockerin käytössä näiden viikkojen aikana.

Docker osaamistani haluaisin syventää yhä enemmän, koska se on nykyään käytössä todella monessa paikkaa ja sen hyödyt ovat kiistattomat. Artikkelissa ”What is Docker and why is it so darn popular” (Vaughan-Nichols 2018.) vapaasti suomennettuna, mikä on Docker ja miksi se on niin suosittu mainitaan, että dockerin mukaan yli 3.5 miljoonaa sovellusta sijoitettu Docker kontteihin ja yli 37 miljardia kontitettua sovellusta on ladattu. Dockerilla pystyy luomaan kaikille kehittäjälle samanlaisen kehitysympäristön ja vaadittavan infrastruktuurin sen ympärille. Dockerin käyttämisessä tuotantoympäristössä on myös monia etuja, kuten sen yhteensopivuus jatkuvan integraation kehityksen kanssa.

Suurimpia haasteita minulle tuotti seurantaviikkojen aikana hiljaisemmat päivät töissä, sekä ajan tasalla olevien lähteiden löytäminen. Välillä oli erittäin haastavaa saada kirjoitettua päiväkirjamerkintöjä, kun kyseessä oli vähän hiljaisempi päivä tai jopa suuri osa viikosta. Suuri haaste oli myös hyvien lähteiden löytäminen, sillä ohjelmointikielet, kirjastot ja tavat kehittyvät valtavaa vauhtia. Tämä aiheuttaa sen, että suuri osa alan kirjallisuudesta on ainakin joiltain osin vanhentunutta jo saapuessaan ulos painosta. Varsinkin front-end tekniikat tuntuvat kehittyvän kovalla vauhdilla eteenpäin, eikä koskaan voi olla varma tuletko käyttämään tämän päivän tekniikoita vaikkapa vuoden päästä.

4.2 Kehittyminen

Olen kehittynyt merkittävästi muutamallakin eri osa-alueella. Olen syventänyt osaamistani Angularin kehittämisessä entisestään, mutta olen myös päässyt kehittämään enemmän

legacy järjestelmiä, mikä on antanut minulle uutta näkemystä sovelluskehittämiseen. On ollut kiinnostavaa huomata, kuinka web sovelluskehitys on mennyt eteenpäin viimeisen noin kymmenen vuoden aikana. Päästessäni työskentelemään niin vanhojen, kuin uudempienkin sovelluksien parissa, olen oppinut huomaamaan monia hyviä puolia modernien web sovelluksien tekemisessä.

Ongelmanratkaisutaitoni on myös ottanut harppauksen eteenpäin. Päästessäni työskentelemään monien erilaisten ongelmien ja sovelluksien parissa olen oppinut löytämään luovempia ratkaisuja käsillä oleviin ongelmiin, sekä tietysti löytänyt uusia lähteitä ratkaisujen hakemiseen. Ennen seurantajakson alkua stackoverflow oli suuressa käytössä ongelmieni ratkaisuun. Nyt osaan käyttää paljon paremmin eri kirjastojen omaa dokumentaatiota tukena löytääkseni ratkaisuja mieltäni vaivaaviin pulmiin. Olen myös entistä paremmin oppinut käyttämään selaimista löytyviä kehittäjä työkaluja päästäkseni ongelmien alkulähteille.

On ollut positiivista huomata, miten olen pystynyt omaksumaan uusia asioita suhteellisen lyhyessäkin ajassa. Olen pystynyt laajentamaan osaamistani ja ymmärrystä yrityksen koko tuotepaletista erittäin paljon näiden seurantaviikkojen aikana. Olen myös uusia asioita omaksuessani pyrkinyt huomioimaan koodin laatua paljon enemmän. Ennen opinnäytetyöni aloittamista päätavoitteeni oli yleensä vain saada toimiva lopputulos, välittämättä niin paljoa siitä, kuinka laadukasta ja helppolukuista itse koodi oli. Palatessani vanhempien sovellusten pariin ja aikaisemmin tekemiini komponentteihin olen huomannut, kuinka vaikeaa niiden sisäistäminen voi olla, jos koodi on monimutkaista ja siihen liittyvät kommentit olemattomia.

4.3 Työn analysointi

Kirjoittaessani päiväkirjamerkintöjä olen pystynyt palaamaan oman tekemiseni äärelle ja peilaamaan sitä eri lähteistä löytyvään materiaaliin. Näin olen saanut vähän isompaa kuvaa siitä, miten asioita muualla tehdään ja omaksumaan uusia toimintatapoja työhöni. Siten olen löytänyt uusia tapoja ongelmien ratkaisuun ja pystynyt kehittämään omaa oppimistani. Olen myös seurantaviikkojen aikana pystynyt kiinnittämään huomiota omaan työskentelyyni ja puuttumaan siitä löytyviin ongelmakohtiin.

4.4 Tulevaisuuden hyödyt

Olen jo nyt pystynyt alkaa hyödyntämään seurantaviikkojen aikana kertyneitä taitoja. Aloitimme töissä uuden projektin, minkä front-endiä olen päässyt rakentamaan täysin alusta alkaen itse. Olen omaksunut paljon uusia asioita seurantaviikkojen aikana tekemistäni huomioista uutta projektia luodessa. Projektin rakenne on alusta alkaen luotu siten, että

sen jatkokehittäminen olisi mahdollisimman loogista ja suoraviivaista. Olen myös siinä päässyt hyödyntämään CSS grid asettelua heti alusta alkaen, mikä tekee sivuston rakenteen luomisesta helppoa. Kyseisessä projektissa pyrin myös minimoimaan käytettävien kolmannen osapuolten kirjastojen määrän, välttääkseni seurantaviikkojen aikana ilmenneitä ongelmia kirjastojen päivitettävyyden kannalta. Projekti on kaikkiaan melkoisen laaja ja ensimmäinen toimiva testiversio olisi tarkoitus saada käyttöön jo kesäkuun loppupuolella. Uskon kuitenkin vakaasti omalta osaltani pystyväni tuohon tavoitteeseen ammentamalla seurantaviikkojen aikana kerryttämääni oppia.

4.5 Jatkokehitysmahdollisuudet

Esimerkiksi uudessa projektissa, jonka töissä aloitin voisi tehdä uuden päiväkirjan, mikä keskittyisi vain kyseiseen projektiin. Tämän avulla olisi mahdollista perehtyä vielä tarkemmin juuri kyseisen projektin toimintamalleihin ja tekniikoihin. Keskittymällä vain yhteen projektiin ja sen toimintamalleihin olisi todennäköisesti mahdollista päästä seuraavalle tasolle taitojeni kehittämisessä.

Nykyisestä opinnäytetyöstäni pystyisin ammentamaan mallia uuden päiväkirjan tekoon ja kehittää siinä hyväksi havaitsemiani tapoja eteenpäin. Oman tekemisen dokumentointi antaa hyvän mahdollisuuden palata aikaisemmin kohtaamieni ongelmien ja niihin kehittämieni ratkaisujen pariin. Niiden pohjalta olisi hyvä päästä takaisin sen hetkiseen ajatusmaailmaan, minkä avulla voisi myös löytää tulevaisuudessa luovia ratkaisuja uusiin ongelmiin.

Lähteet

Angular 2018. Style Guide. Luettavissa: https://angular.io/guide/styleguide. Luettu: 19.05.2018.

Feathers, M. 2005. Working effectively with legacy code. Prentice Hall. Upper Saddle River, NJ.

House, C 2018. A Complete Guide to Grid 2018. Luettavissa: https://css-tricks.com/snippets/css/complete-guide-grid/. Luettu 21.05.2018.

Hämäläinen, P 2016. Mikropalvelut nousivat hypen huipulle – mitä hyötyä niistä on? Luettavissa: https://www.tivi.fi/Kaikki_uutiset/mikropalvelut-nousivat-hypen-huipulle-mita-hyotya-niista-on-6534379. Luettu: 27.05.2018

Learn RxJS 2018. forkJoin. Luettavissa: https://www.learnrxjs.io/operators/combination/forkjoin.html. Luettu: 15.02.2018.

Martin, R. 2009. Clean Code A Handbook of Agile Software Craftsmanship. Prentice Hall. Upper Saddle River, NJ.

Martin, R. 2017. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall. Upper Saddle River, NJ.

Matthias, K & Kane, S. 2015. Docker: Up and Running. O'Reilly Media Inc. Sebastopol.

Resig, J. Bibeault, B. & Maras, J. 2016. Secrets of the JavaScript Ninja, Second Edition. Manning Publications Co. Shelter Island.

Santacroce, F. Olsson, A. Voss, R. & Narębski, J. 2016. Git: Mastering Version Control. Packt Publishing Ltd. Birmingham.

Vaughan-Nichols, J 2018. What is docker and why is it so darn popular. Luettavissa: https://www.zdnet.com/article/what-is-docker-and-why-is-it-so-darn-popular/. Luettu 21.05.2018.